INVITED PAPER

Time-Varying Convex Optimization: Time-Structured Algorithms and Applications

This article reviews a broad class of algorithms for time-varying optimization with an emphasis on both algorithmic development and performance analysis.

By Andrea Simonetto, *Member IEEE*, Emiliano Dall'Anese, *Member IEEE*, Santiago Paternain, *Member IEEE*, Geert Leus, *Fellow IEEE*, and Georgios B. Giannakis, *Fellow IEEE*

ABSTRACT | Optimization underpins many of the challenges that science and technology face on a daily basis. Recent years have witnessed a major shift from traditional optimization paradigms grounded on batch algorithms for medium-scale problems to challenging dynamic, time-varying, and even huge-size settings. This is driven by technological transformations that converted infrastructural and social platforms into complex and dynamic networked systems with even pervasive sensing and computing capabilities. This article reviews a broad class of state-of-the-art algorithms for time-varying optimization, with an eye to performing both algorithmic development and performance analysis. It offers a comprehensive overview of available tools and methods and unveils open challenges in application domains of broad range of interest. The real-world examples presented include smart power systems, robotics, machine learning, and data analytics, highlighting domain-specific issues and solutions. The ultimate goal is to exemplify wide engineering relevance of analytical tools and pertinent theoretical foundations.

KEYWORDS | Convergence of numerical methods; optimization methods.

I. INTRODUCTION

Optimization is prevalent across many engineering and science domains. Tools and algorithms from convex optimization have been traditionally utilized to support a gamut of data-processing, monitoring, and control tasks across areas as diverse as communication systems, power and transportation networks, medical and aerospace engineering, video surveillance, and robotics, just to name a few. Recently, some of these areas—and, in particular, infrastructures such as power, transportation, and communication networks, as well as social and e-commerce platforms—are undergoing a foundational transformation, driven by major technological advances across various sectors, the information explosion propelled by online social media, and pervasive sensing and computing capabilities. Effectively, these infrastructures and platforms are revamped into complex systems operating in highly dynamic environments and with high volumes of heterogeneous information. This calls for revisiting several facets of workhorse optimization tools and methods under a different lens: the ability to process data streams and provide decision-making capabilities at time scales that match the dynamics of the underlying physical, social, and engineered systems using solutions that are grounded on

Manuscript received December 14, 2019; revised March 31, 2020; accepted June 9, 2020. Date of publication July 3, 2020; date of current version October 27, 2020. The work of Emiliano Dall'Anese was supported in part by the National Science Foundation (NSF) under Grant 1941896. The work of Georgios B. Giannakis was supported in part by NSF under Grant 1711471 and Grant 1901134. *(Corresponding author: Andrea Simonetto.)*

Andrea Simonetto is with the Optimization and Control Group, IBM Research Ireland, Dublin 15, Ireland (e-mail: andrea.simonetto@ibm.com).

Emiliano Dall'Anese is with the College of Engineering and Applied Science, University of Colorado Boulder, Boulder, CO 80309 USA (e-mail: emiliano.dallanese@colorado.edu).

Santiago Paternain is with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104 USA (e-mail: spater@seas.upenn.edu).

Geert Leus is with the Faculty of Electrical, Mathematics and Computer Science, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: g.j.t.leus@tudelft.nl).

Georgios B. Giannakis is with the Digital Technology Center, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: georgios@umn.edu).

Digital Object Identifier 10.1109/JPROC.2020.3003156

conventional optimization methods can no longer be taken for granted. Let us consider power grids, as a representative example: economic optimization at the network level was performed using batch solvers at the minute or hour level to optimally dispatch large-scale fossil-fuel generation based on predictable loads; on the other hand, novel optimization tools are now desirable to carry network optimization tasks with solvers capable of coping with volatile renewable generation while managing the operation of a massive number of distributed energy resources (DERs). These considerations have spurred research and engineering efforts that are centered around time-varying optimization, a formalism for modeling and solving optimization tasks in engineering and science under dynamic environments.

Continuously varying optimization problems represent a natural extension of time-invariant programs when the cost function and constraints may change continuously over time [1]–[4]. Recently, time-varying optimization formalisms and the accompanying online solvers have been proposed both in continuous-time [5], [6] and in discrete-time settings [7], [8]. Their main goal is to develop algorithms that can track trajectories of the optimizers of the continuously varying optimization program (up to asymptotic error bounds). The resultant algorithmic frameworks have demonstrated reliable performance in terms of convergence rates (CRs) with error bounds that relate tracking capabilities with computational complexity; these features make time-varying algorithms an appealing candidate to tackle dynamic optimization tasks at scale, across many engineering and science domains.

This article overviews key modeling and algorithmic design concepts, with emphasis on textittime-structured (structured for short) time-varying algorithms for convex time-varying optimization. The term "structured" here refers to algorithms that take advantage of the inherent temporal structure, meaning they leverage prior information (such as Lipschitz continuity or smoothness) on the evolution of the optimal trajectory to enhance convergence and tracking. In contrast, the term "unstructured" refers to time-varying algorithms that simply rely on current information of cost and constraints. This also differentiates the present "time-structured" class from interactive algorithms (that belong to the unstructured class), which are tailored to learner-environment or player-environment settings; for example, the popular online convex optimization (OCO) setup [9], where online algorithms decide on current iterates (using only information of past cost functions), and subsequently the environment reveals partial or full information about the function to be optimized next.

Fig. 1 depicts a typical time-varying optimization setting. Streaming data are generated from time-varying systems, as in renewable generation that is intermittent, traffic conditions that change in transportation systems, or drop-off points for drone delivery that are mobile.

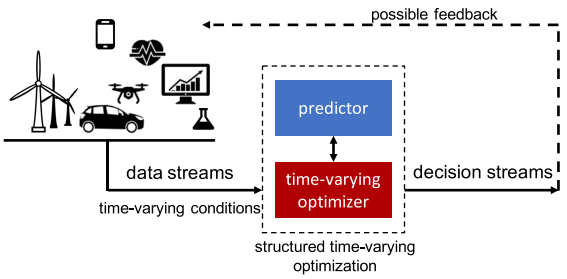

Fig. 1. ***Setup of time-varying optimization algorithms. Streaming data generated by time-varying systems are input to an optimizer. The optimizer can employ a predictor (that could be an oracle or a well-defined model), which feeds the optimizer with predictions of how the optimization problem will change. The optimizer then delivers a decision stream that is used to take actions that could be possibly fed back to affect the dynamical system operation.***

Such settings inherit time variability in the optimization problem at hand. The optimizer can leverage a predictor (an oracle or a well-defined model), which feeds the optimizer with predictions of how the optimization problem may evolve over time. The optimizer then delivers a decision stream (i.e., an approximate optimizer) that is used to take operational actions such as committing a generator, or, adopting an optimal routing schedule for ridesharing vehicles. These actions could also affect and are therefore fed back to the system (e.g., the optimal ridesharing schedule alters traffic and availability of vehicles in the future). When the input data streams are of large scale and/or the decisions need to be made at a high frequency, traditional batch algorithms (that exactly solve the optimization problem at each time) are not viable because of underlying computational complexity bottlenecks. Hence, an online computationally frugal optimization becomes essential to produce solutions in a timely fashion.

To further motivate structured time-varying methods, Fig. 2[1] illustrates the asymptotic tracking error (asymptotic difference between optimal decisions and decisions delivered by some algorithms that will be described shortly) for different sampling periods (h) of discrete-time algorithms, for a robot-tracking problem (see [10] for the setting). The value of exploiting the temporal structure of the problem can be appreciated. Even keeping computational time fixed, structured algorithms outperform unstructured ones (here by several orders of magnitude). Exploiting this structure may lead to a reduction of the computational cost of the algorithms. This is the case, for instance, when using model predictive control (MPC) on the Hicks reactor [12] (see Fig. 3 adapted from [13]).

[1]Unstructured algorithms 0, 1, and 2 are, in this case, online versions of the proximal gradient method, for which we perform 5, 7, and 9 passes of the methods, respectively. Structured algorithms here employ either a first- or a second-order Taylor model (for structured 1 and 2, respectively), and 5 and 20 passes of an online version of the proximal gradient method on a simplified quadratic problem (see [11] for further details).

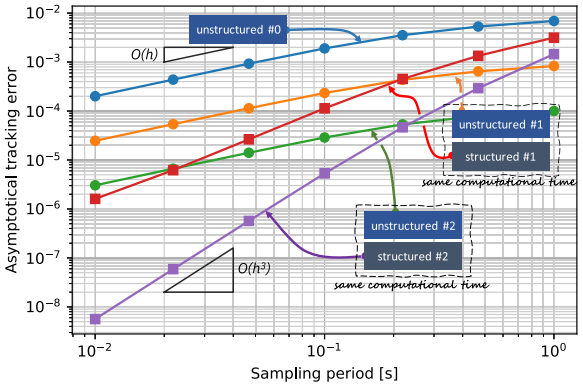

Fig. 2. ***Structured algorithms can outperform unstructured ones, even keeping the computational time fixed: here for a robot-tracking problem. See text and footnote for description of the algorithms and [10], [11] for the problem setting. This figure will be referred to multiple times in this article and the different elements will be clarified.***

The main goal of this overview article is threefold:

1) to expose models and algorithms for structured time-varying optimization settings, from both analytical and an application-oriented perspectives;
2) to demonstrate applications of structured time-varying optimization algorithms (and deep dive into two, namely a robotic and a power system application);
3) to draw links with the growing landscape of unstructured algorithms for dynamic optimization problems.

Setting and notation: We deal with convex optimization problems [14], [15], as well as first-order algorithms [16]. Vectors are represented with $\boldsymbol{x} \in \mathbb{R}^n$, and the Euclidean norm is denoted as $\|\cdot\|$. We mainly deal with strongly convex and smooth functions. A function $f : \mathbb{R}^n \to \mathbb{R}$ is m-strongly convex for a constant $m > 0$, that is, $f(\boldsymbol{x}) - m/2\|\boldsymbol{x}\|^2$ is convex, and L-smooth for a constant $L > 0$ iff its gradient is L-Lipschitz continuous or equivalently iff $f(\boldsymbol{x}) - L/2\|\boldsymbol{x}\|^2$ is concave. Sometimes, we deal with extended real-valued functions $\varphi : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ (which can explicitly admit infinite values, e.g., the indicator function). We define the subdifferential of φ as the set $\boldsymbol{x} \mapsto \{\boldsymbol{z} \in \mathbb{R}^n \mid \forall \boldsymbol{y} \in \mathbb{R}^n : \langle \boldsymbol{y} - \boldsymbol{x}, \boldsymbol{z}\rangle + \varphi(\boldsymbol{x}) \leq \varphi(\boldsymbol{y})\}$. Given a convex set $\mathcal{X}$, $\text{proj}_{\mathcal{X}}\{\boldsymbol{x}\}$ denotes a closest point to $\boldsymbol{x}$ in $\mathcal{X}$, namely $\text{proj}_{\mathcal{X}}\{\boldsymbol{x}\} \in \arg\min_{\boldsymbol{y}\in\mathcal{X}} \|\boldsymbol{x} - \boldsymbol{y}\|$. We also use $O(\cdot)$ to represent the big-O notation.

II. TIME-VARYING OPTIMIZATION

Let $f : \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R}$ be a convex function parametrized over time, that is, $f(\boldsymbol{x}; t)$, where $\boldsymbol{x} \in \mathbb{R}^n$ is the decision variable and $t \geq 0$ is time. Let $X(t) \subseteq \mathbb{R}^n$ be a convex set, which may also change over time. We are interested here in solving

$$\min_{\boldsymbol{x}\in X(t)} f(\boldsymbol{x}; t), \quad \text{for all } t \geq 0. \tag{1}$$

To simplify exposition, we assume that the cost function f is m-strongly convex for all t (this is nevertheless a standard assumption in most prior works) and that the constraint set is never empty. With these assumptions in place, at any time t, Problem (1) has a unique global optimizer. This translates to finding the optimal solution trajectory

$$\boldsymbol{x}^\star(t) := \operatorname*{argmin}_{\boldsymbol{x}\in X(t)} f(\boldsymbol{x}; t), \quad \text{for all } t \geq 0. \tag{2}$$

As an example, for the robot-tracking problem for which the results have been shown in Fig. 2, $f(\boldsymbol{x}; t)$ is a time-varying performance metric for the tracking performance of a robot formation that is following a robot leader, for example, $f(\boldsymbol{x}; t) = \|\boldsymbol{x} - \boldsymbol{b}(t)\|^2 + \mathcal{R}(\boldsymbol{x})$, where $\mathcal{R}(\boldsymbol{x})$ is some pertinent regularization function and $\boldsymbol{b}(t)$ encodes the tracking signal. On the other hand, $X(t)$ represents some physical or hardware constraints for the robots. At each t', the information available is $\{f(\boldsymbol{x}; t), t \leq t'\}$ and $\{X(t), t \leq t'\}$; on the basis of a possibly limited computational complexity, and without any information regarding future costs and constraints, the next decision $\boldsymbol{x}(t')$ has to be made; the objective is to produce a decision $\boldsymbol{x}(t')$ that is as close as possible to $\boldsymbol{x}^\star(t')$.

If Problem (2) changes slowly and sufficient computational power is available, existing batch optimization methods may identify the optimal trajectory $\boldsymbol{x}^\star(t)$; for example, if the parameter $\boldsymbol{b}(t)$ mentioned above exhibits step changes every 10 s, and a distributed batch algorithm converges in 5 s, then $\boldsymbol{x}^\star(t)$ can be identified (within a given accuracy). On the other hand, in highly dynamic settings, computational and communication bottlenecks may prevent batch methods to produce solutions in a timely manner [e.g., $\boldsymbol{b}(t)$ changes every 0.5 s, and a distributed batch algorithm converges in 5 s]; the problem then becomes related to the synthesis of computationally affordable algorithms that can produce an approximate optimizer trajectory $\hat{\boldsymbol{x}}(t)$ on the fly; accordingly, a key performance of these algorithms is the "distance"

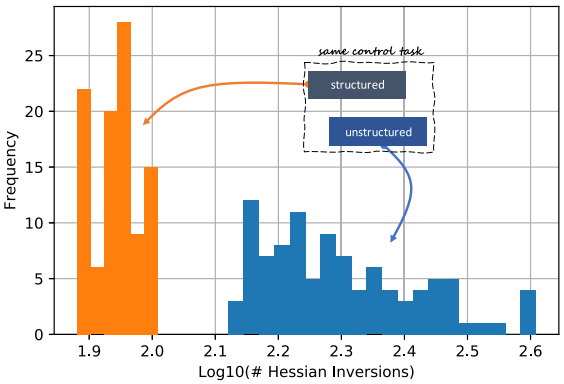

Fig. 3. ***Histograms with the total number of Hessian inversions required to control the Hicks reactor [12] for structured and unstructured MPC solvers. Exploiting the temporal structure reduces the computational complexity, measured by the number of inversions of the Hessian, in solving the MPC.***

between the approximate solution trajectory $\hat{\boldsymbol{x}}(t)$ and the optimal one $\boldsymbol{x}^\star(t)$.

A. Time-Structured and Time-Unstructured Algorithms

The term "structured" refers to algorithms that, at time t, exploit a (learned) model to predict how the optimizer trajectory $\hat{\boldsymbol{x}}(t)$ evolves, say from t to t', and then correct the prediction by approximately solving the optimization problem obtained at t'. Unstructured algorithms instead have no evolution model and use only the optimization problems that are revealed at each time. A useful parallelism is the Kalman filter versus the recursive least-squares (RLSs) estimator. Although the Kalman filter is endowed with a model to predict how the state evolves in time and then corrects the prediction with new up-to-date observations, RLS relies solely on the observations. Structured time-varying algorithms leverage an evolution model to predict and observe new problems to correct their predictions. Unstructured ones rely only on observations.

B. Performance Metrics

Different performance metrics can be considered for online algorithms that generate approximate trajectories for Problem (2). They all capture the fact that the computation of $\hat{\boldsymbol{x}}(t)$ is time-limited, computationally limited, or both, and therefore $\hat{\boldsymbol{x}}(t)$ is an approximate optimizer at time t. Here, it is more fruitful to look at the computation of $\hat{\boldsymbol{x}}(t)$ as limited by time: to compute $\hat{\boldsymbol{x}}(t)$, one has at most Δt.

An immediate performance metric is the asymptotical tracking error (ATE), defined as

$$\text{ATE} := \limsup_{t\to\infty} \|\hat{\boldsymbol{x}}(t) - \boldsymbol{x}^\star(t)\| \tag{3}$$

which captures how the algorithm performs in an asymptotic sense. In general, one seeks asymptotic consistency of the algorithm, that is, if $\boldsymbol{x}^\star(t)$ is asymptotically stationary, then the ATE should be zero. However, if $\boldsymbol{x}^\star(t)$ is time-varying, the ATE cannot be zero for unstructured algorithms, while it could be zero for structured algorithms.[2]

A second metric that is relevant for time-varying optimization problems is the time rate (TR), defined as

$$\text{TR} := \frac{\text{time required for the computation of } \hat{\boldsymbol{x}}(t)}{\text{time allowed for the computation of } \hat{\boldsymbol{x}}(t)}. \tag{4}$$

Here, we define "time required," as the time needed for the computation of an approximate $\hat{\boldsymbol{x}}(t)$, which delivers a predefined ATE. TR is a key differentiator for time-varying optimization: online algorithms need to be able to deliver an approximate $\hat{\boldsymbol{x}}(t)$ in the allocated time. Data streams generate decision streams with the same frequency, and the online optimization algorithm needs to have a TR less than 1 to be implementable. The TR also sets an important tradeoff between ATE and implementability. One typically cannot expect a very low ATE and implementable solutions.

The third metric is the CR, which can be informally defined as

$$\text{CR} := \text{"how fast" an algorithm converges to the ATE.} \tag{5}$$

CR will be formalized for discrete-time algorithms and continuous-time algorithms shortly. For discrete-time algorithms, under current modeling assumptions, it will be possible to derive Q-linear convergence results (definition given later on); on the other hand, for continuous-time algorithms, the CR will be exponential and related to the exponent of a carefully constructed Lyapunov function.

Typically, the algorithmic design will involve a tradeoff between the ATE and CR; for instance, lower levels of ATE may be achievable at the expense of a higher CR. CR is then important, not only at the start, but also when abrupt changes happen (and then the CR captures how fast the algorithm responds to those changes and disturbances).

An additional metric is a measure that distinguishes between the structured and unstructured algorithms, here referred to as structure gain (SG). It could be defined as the ratio between the ATE obtained with a structured method divided by the ATE obtained with a competing unstructured method, that is,

$$\text{SG} := \frac{\text{ATE for selected structured method}}{\text{ATE for competing unstructured method}}. \tag{6}$$

Of course, both algorithms are constrained to use the same computational time for $\hat{\boldsymbol{x}}(t)$. This metric assists in the decision as to whether to use the selected structured or the competing unstructured algorithm for a given time-varying optimization task. We have already seen in Fig. 2 that the value of structure can lead to an SG greater than 1, further motivating the use of structured methods.

In Fig. 4, a general overview of the algorithms that will be presented in this article is given together with their connections.

C. Discrete-Time Algorithms

This section surveys discrete-time algorithms. Consider sampling Problem (2) at defined sampling times $\{t_k = kh, k \in \mathbb{N}\}$, with h being the sampling period; thus, one arrives at a sequence of time-invariant problems

$$\boldsymbol{x}^\star(t_k) := \operatorname*{argmin}_{\boldsymbol{x}\in X} f(\boldsymbol{x}; t_k), \quad t_k = kh, k \in \mathbb{N}. \tag{7}$$

For simplicity of exposition, we drop the time dependence of the constraints and consider static sets. As long as one

[2] A dynamic regret notion based on the cost function is also available, but we do not discuss this here. The interested reader is referred to [17] and [18].

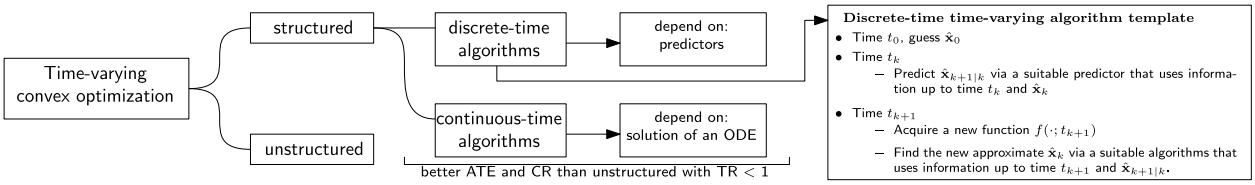

Fig. 4. ***Algorithms presented in this article.***

can solve each (time-invariant) Problem (7) within an interval h using existing algorithms, then a "batch solution mode" is sufficient to identify the optimal trajectory $\{\boldsymbol{x}^\star(t_k), k \in \mathbb{N}\}$. This batch approach is, however, hardly viable, except for low-dimensional problems that can be sampled with sufficiently long sampling periods (i.e., when the problem changes sufficiently slowly). We focus here on the case where one can afford only one or a few steps of a given algorithm within an interval h, that is, an online approach. This setting can then be cast as the problem of synthesizing online algorithms that can track $\{\boldsymbol{x}^\star(t_k), k \in \mathbb{N}\}$, within a given ATE.

A key assumption for any online approach is that the difference between solutions at two consecutive times is bounded.

Assumption 1: The distance between optimizers at subsequent times is uniformly upper bounded as

$$\|\boldsymbol{x}^\star(t_k) - \boldsymbol{x}^\star(t_{k-1})\| \leq K, \quad \forall k > 0,\ K < \infty.$$

The constant K will play a key role in the ATE, as shown shortly. Assumption 1 is general, inasmuch it does not forbid the underlying trajectory $\boldsymbol{x}^\star(t)$ to have finite jumps.[3]

A stronger assumption, often required in time-structured optimization, is that the time derivative of the gradient of the cost function,[4] that is, $\nabla_{t\boldsymbol{x}} f(\boldsymbol{x};t)$, is bounded.

Assumption 2: For all t and all $\boldsymbol{x}$: $\|\nabla_{t\boldsymbol{x}} f(\boldsymbol{x};t)\| \leq \Delta_0 < \infty$.

Assumption 2, along with m-strong convexity of the cost function, guarantees that the trajectory $\boldsymbol{x}^\star(t)$ is globally Lipschitz in time [19], [20], and in particular

$$\|\boldsymbol{x}^\star(t') - \boldsymbol{x}^\star(t)\| \leq \frac{\Delta_0}{m}|t' - t|. \tag{8}$$

This is key for structured time-varying algorithms and typically not required in unstructured algorithms or in OCO [9]. Note further that Assumption 2 implies Assumption 1 with the choice $K = \Delta_0\, h/m$.

In this discrete-time setting, an online algorithm will generate a sequence of approximate optimizers. Hereafter, we will denote the output of the algorithm at time t_k for simplicity as $\hat{\boldsymbol{x}}_k$, while we denote the sequence as $(\hat{\boldsymbol{x}}_k)_{k\in\mathbb{N}_+}$. Different algorithms will be distinguished based on which predictor they use and how they generate $\hat{\boldsymbol{x}}_k$.

1) No-Predictor Algorithms: In this case, online algorithms do not have a "prediction" step; rather, they only perform "corrective" steps once the cost function is acquired. These algorithms are called in different ways (among which catching up, running, correction-only, and unstructured) and probably firstly appeared with Moreau [1]. For example, a running projected gradient to approximately solve (7) is given by the recursion

$$\hat{\boldsymbol{x}}_0 = \mathbf{0}, \quad \hat{\boldsymbol{x}}_k = \text{proj}_X\{\hat{\boldsymbol{x}}_{k-1} - \alpha\nabla_{\boldsymbol{x}} f(\hat{\boldsymbol{x}}_{k-1};t_k)\},\ k \in \mathbb{N} \tag{9}$$

where $\text{proj}_X\{\cdot\}$ denotes the projection operator and α is a carefully chosen step size (that could be time-varying as well). In (9), the projected gradient is applied once per time step k, but one could also apply multiple gradient steps, say C, per time step. Notwithstanding this, in general, these unstructured discrete-time algorithms achieve a high ATE. To formalize this result, we focus on a class of algorithms that exhibit a Q-linear convergence. In particular, let $\mathcal{M}$ be an algorithm that when applied to $\hat{\boldsymbol{x}}_k$ at time t_{k+1} for function $f(\boldsymbol{x};t_{k+1})$ produces an $\hat{\boldsymbol{x}}_{k+1}$ for which

$$\|\hat{\boldsymbol{x}}_{k+1} - \boldsymbol{x}^\star(t_{k+1})\| \leq \varrho\|\hat{\boldsymbol{x}}_k - \boldsymbol{x}^\star(t_{k+1})\|, \quad \varrho \in (0,1) \tag{10}$$

then algorithm $\mathcal{M}$ is called Q-linear convergent. This class is common in time-varying optimization (e.g., projected gradient (9) is Q-linear on a m-strongly convex and L-smooth cost function [16]). When the algorithm $\mathcal{M}$ is then applied C times [as e.g., in (30)], we obtain $\|\hat{\boldsymbol{x}}_{k+1} - \boldsymbol{x}^\star(t_{k+1})\| \leq \varrho^C\|\hat{\boldsymbol{x}}_k - \boldsymbol{x}^\star(t_{k+1})\|$. The following general result is in place.

Theorem 1 (Informal): Let $\mathcal{M}$ be an optimization algorithm that converges Q-linearly as in (10). Then, under Assumption 1, the same algorithm $\mathcal{M}$ applied C times for each time t_k, converges Q-linearly to the optimizer trajectory of a time-varying problem up to an error bound as

$$\|\hat{\boldsymbol{x}}_{k+1} - \boldsymbol{x}^\star(t_{k+1})\| \leq \varrho^C(\|\hat{\boldsymbol{x}}_k - \boldsymbol{x}^\star(t_k)\| + K)$$

and $\limsup_{k\to\infty}\|\hat{\boldsymbol{x}}_k - \boldsymbol{x}^\star(t_k)\| = \varrho^C O(K) = (\Delta_0/m)\varrho^C O(h)$ where the last equality is valid under Assumption 2.

[3]Meaning that $\boldsymbol{x}^\star(t)$ can be discontinuous in time, but the discontinuity has to be bounded, so that Assumption 1 holds for the choice of sampling period.

[4]This can be generalized for a nonsmooth cost function of the form $f(\boldsymbol{x};t) + g(\boldsymbol{x})$, as long as $f(\boldsymbol{x};t)$ is differentiable, e.g., $\|x - t\|^2 + |x|$ [11].

Proof (Sketch): At time t_k, if algorithm $\mathcal{M}$ is applied C times, starting on $\hat{\boldsymbol{x}}_k$ and ending at $\hat{\boldsymbol{x}}_{k+1}$, by Q-linear convergence of $\mathcal{M}$, we can write

$$\begin{aligned}&\|\hat{\boldsymbol{x}}_{k+1} - \boldsymbol{x}^\star(t_{k+1})\| \\ &\quad \leq \varrho^C(\|\hat{\boldsymbol{x}}_k - \boldsymbol{x}^\star(t_{k+1})\|) \\ &\quad \leq \varrho^C(\|\hat{\boldsymbol{x}}_k - \boldsymbol{x}^\star(t_k)\| + \|\boldsymbol{x}^\star(t_{k+1}) - \boldsymbol{x}^\star(t_k)\|)\end{aligned}$$

and by using Assumption 1 the first claim is established. The second claim is proven by recursively applying the first claim, and by geometric series summation. ■

The results of the theorem are general and assert that the sequence $(\hat{\boldsymbol{x}}_k)$ tracks the solution trajectory up to a ball of size $\varrho^C O(K)$. If $C \to \infty$, the time-invariant problem is solved exactly and we are back to the batch mode (and the error is 0), that is, the time-varying algorithm is asymptotically consistent. If Assumption 2 holds true, then the asymptotic error is proportional to the sampling period h (see Fig. 2). In addition, for fixed $\varrho \in (0,1)$, $C < \infty$, and if the path-length $\sum_{k=1}^{T} \|\boldsymbol{x}^\star(t_k) - \boldsymbol{x}^\star(t_{k-1})\|$ grows at least linearly in T, no unstructured method of this type can reach a zero ATE [21], [22].

2) Predictors: We now focus on discrete-time algorithms that are endowed with a prediction. Various predictors are considered, and we will call as $\hat{\boldsymbol{x}}_{k+1|k}$ the predicted decision variable for time t_{k+1} with information up to time t_k.

3) Clairvoyant Oracles and Expert Oracles: Clairvoyant oracles offer an exact prediction, that is, they provide a $\hat{\boldsymbol{x}}_{k+1|k}$, for which $\|\hat{\boldsymbol{x}}_k - \boldsymbol{x}^\star(t_k)\| = \|\hat{\boldsymbol{x}}_{k+1|k} - \boldsymbol{x}^\star(t_{k+1})\|$, as if they knew the function $f(\cdot; t_{k+1})$ and its gradient at time t_k. In this context, clairvoyant oracles completely remove the time effect in the optimizer and the optimizer can proceed as if the cost function were not varying in time. Clairvoyant oracles are impractical (they need to have a perfect knowledge of the future), but they offer good performance lower bounds (since one cannot do better than them). A noteworthy example of when one can use a clairvoyant oracle is when the cost function has a time drift, that is, $f(\boldsymbol{x}; t) = f(\boldsymbol{x} + \boldsymbol{\alpha} t)$, and the oracle can estimate the drift vector $\boldsymbol{\alpha}$ exactly based on historical data.

Expert oracles, hints, or predictable sequences are considered, for example, in [23]–[25]. In [24], one has access to a sequence $(\boldsymbol{m}_k)_{k\in\mathbb{N}_+}$ of gradient approximators. When $\boldsymbol{m}_k = \boldsymbol{0}$, that is, meaning no knowledge or prediction about the future, we recover an unstructured algorithm. When $\boldsymbol{m}_k = \nabla_{\boldsymbol{x}} f(\boldsymbol{x}; t_k)$ at time t_k, then one recovers the online algorithm of [26]. Finally, when $\boldsymbol{m}_k = \nabla_{\boldsymbol{x}} f(\boldsymbol{x}; t_{k+1})$, one recovers a clairvoyant oracle. Based on the error $\|\boldsymbol{m}_k - \nabla_{\boldsymbol{x}} f(\boldsymbol{x}; t_{k+1})\|$, one can then derive dynamic ATE results.

4) Model-Based Predictors: These predictors are built on a model of the variations of the cost function, or of its parameters.

a) Prediction based on first-order optimality conditions [4], [7], [8], [27]–[29]: A large class of predictors comes from deriving models based on first-order optimality conditions. We could call these predictors environment-agnostic, since they are not interested in modeling how the environment changes, but only how the optimization problem is affected. To introduce these predictors, let us consider an unconstrained problem [easier than Problem (7)] as

$$\boldsymbol{x}^\star(t_k) = \operatorname*{argmin}_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}; t_k). \tag{11}$$

To derive a model for how the problem is changing from t_k to t_{k+1}, we look at the first-order optimality conditions at time t_k, which can be framed as

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}; t_k) = \boldsymbol{0}. \tag{12}$$

To predict, how this first-order optimality condition changes in time, with information available up to t_k, we use a Taylor expansion around $(\hat{\boldsymbol{x}}_k; t_k)$ as

$$\begin{aligned}\boldsymbol{0} = \nabla_{\boldsymbol{x}} f(\boldsymbol{x}; t_{k+1}) \approx \varphi_k(\boldsymbol{x}) := \nabla_{\boldsymbol{x}} f(\hat{\boldsymbol{x}}_k; t_k) \\ + \nabla_{\boldsymbol{x}\boldsymbol{x}} f(\hat{\boldsymbol{x}}_k; t_k)(\boldsymbol{x} - \hat{\boldsymbol{x}}_k) + h\, \nabla_{t\boldsymbol{x}} f(\hat{\boldsymbol{x}}_k; t_k)\end{aligned} \tag{13}$$

where it is assumed that the Hessian $\nabla_{\boldsymbol{x}\boldsymbol{x}} f(\hat{\boldsymbol{x}}_k; t_k)$ exists, as well as the time derivative of the gradient $\nabla_{t\boldsymbol{x}} f(\hat{\boldsymbol{x}}_k; t_k)$, leading to the prediction model[5]

$$\begin{aligned}\varphi_k(\hat{\boldsymbol{x}}_{k+1|k}) = \boldsymbol{0} \implies \\ \hat{\boldsymbol{x}}_{k+1|k} = \hat{\boldsymbol{x}}_k - \nabla^{-1}_{\boldsymbol{x}\boldsymbol{x}} f(\hat{\boldsymbol{x}}_k; t_k) \\ \times [\nabla_{\boldsymbol{x}} f(\boldsymbol{x}; t_k) + h\, \nabla_{t\boldsymbol{x}} f(\hat{\boldsymbol{x}}_k; t_k)].\end{aligned} \tag{14}$$

The prediction (14) represents a nonlinear discrete-time model to compute $\boldsymbol{x}_{k+1|k}$. Note that $\varphi_k(\boldsymbol{x})$ can be interpreted as a specific choice for the gradient approximator $\boldsymbol{m}_k$ in [24] (see the discussion in the oracles paragraph). Let us now consider a slightly more general setting than Problem (7) as

$$\boldsymbol{x}^\star(t_k) = \operatorname*{argmin}_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}; t_k) + g(\boldsymbol{x}) \tag{15}$$

where $g : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is a convex closed and proper function (e.g., $g(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$). Problem (7) is a special case of (15), when $g(\boldsymbol{x})$ is the indicator function of the set X. Once again, we look at the first-order optimality conditions at time t_k, which can be framed as the generalized equation [19]

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}; t_k) + \partial g(\boldsymbol{x}) \ni \boldsymbol{0}. \tag{16}$$

[5]The time derivative $\nabla_{t\boldsymbol{x}} f(\boldsymbol{x}; t_k)$ can be obtained via first-order backward finite difference, if not available otherwise (see [20] and [27]).

To predict how this first-order optimality condition changes in time (with information up to t_k), one can use a Taylor expansion around $(\hat{\boldsymbol{x}}_k; t_k)$, leading to the prediction model

$$\varphi_k(\hat{\boldsymbol{x}}_{k+1|k}) + \partial g(\hat{\boldsymbol{x}}_{k+1|k}) \ni \mathbf{0}. \tag{17}$$

Thus, the prediction step requires the solution of this approximated generalized equation with initial condition $\hat{\boldsymbol{x}}_k$, which can be obtained, or approximated, cheaply with, for example, a few passes of a proximal gradient method [30] (cheaply since φ_k is a quadratic function). The formulation (17) represents the prediction model for the presented class of optimization problems (15), for a first-order Taylor expansion. Other prediction models exist for other classes of optimization problems [27], [28], for higher-order Taylor expansions [4], and for more complex numerical integration methods [29], [31]–[34].

b) Prediction based on parameter estimation [35]: When the time dependence hides a parameter dependence, then models obtained via filtering are a viable alternative. Let $\boldsymbol{b}(t) \in \mathbb{R}^l$ be a parameter, and let the function $f(\boldsymbol{x}; t) = f(\boldsymbol{x}; \boldsymbol{b}(t))$: for example, the cost depends on the data stream $\boldsymbol{b}(t)$ representing, for example, the position of a robot to track. Then $\boldsymbol{b}(t)$ at time t_{k+1} can be estimated via, for example, a Kalman filter based on the linear time-invariant model

$$\boldsymbol{b}(t_{k+1}) = \boldsymbol{\Gamma}\boldsymbol{b}(t_k) + \boldsymbol{w}_k, \quad \boldsymbol{y}_k = \boldsymbol{\Phi}\boldsymbol{b}(t_k) + \boldsymbol{n}_k \tag{18}$$

for the given matrices $\boldsymbol{\Gamma} \in \mathbb{R}^{l\times l}$, $\boldsymbol{\Phi} \in \mathbb{R}^{q\times l}$, observations $\boldsymbol{y}_k \in \mathbb{R}^q$, and noise terms $\boldsymbol{w}_k \in \mathbb{R}^l, \boldsymbol{n}_k \in \mathbb{R}^q$. Then the prediction model requires the (approximate) solution of the problem

$$\hat{\boldsymbol{x}}_{k+1|k} \approx \operatorname*{argmin}_{\boldsymbol{x}\in X} f(\boldsymbol{x}; \hat{\boldsymbol{b}}_{k+1}) \tag{19}$$

with $\hat{\boldsymbol{b}}_{k+1}$ being the forecast $\boldsymbol{b}(t_{k+1})$ based on the model (18) via, for example, a Kalman filter. Other models can be thought of based on nonlinear models, more complicated forecasters, and even neural networks.

5) Prediction–Correction Algorithms: We have presented a few predictors for discrete-time time-varying optimization algorithms. No general result exists to encompass all the predictors. However, for a particular class of predictors (the one that employs first-order optimality conditions as the prediction model), some general results can be derived. These methods are known as prediction–correction methods (since they predict how the optimization problem changes and then they correct for the errors in predictions based on the newly acquired cost [8], [27]) and have roots in nonstationary optimization [2], [36], parametric programming [4], [7], [19], [37], and continuation methods in numerical mathematics [38].

Consider Problem (15) for simplicity (although arguments are generalizable). Let $\mathcal{P}$ be a predictor method that approximates $\hat{\boldsymbol{x}}_{k+1|k}$ based on (17), in a Q-linear convergent fashion: one application of $\mathcal{P}$ acting on $\hat{\boldsymbol{x}}_k$ delivers a $\hat{\boldsymbol{x}}'_{k+1|k}$ for which

$$\|\hat{\boldsymbol{x}}'_{k+1|k} - \hat{\boldsymbol{x}}_{k+1|k}\| \leq \varrho_1 \|\hat{\boldsymbol{x}}_k - \hat{\boldsymbol{x}}_{k+1|k}\|, \quad \varrho_1 \in (0,1). \tag{20}$$

For example, $\mathcal{P}$ could be a proximal gradient algorithm, in which case

$$\hat{\boldsymbol{x}}'_{k+1|k} = \operatorname{prox}_{\alpha g}\{\hat{\boldsymbol{x}}_k - \alpha\nabla_{\boldsymbol{x}}\varphi_k(\hat{\boldsymbol{x}}_k)\} \tag{21}$$

where $\operatorname{prox}_{\alpha g}\{\cdot\}$ is the proximal operator for function g and step-size α, which could be applied one or multiple, say P, times for time step. Let now $\mathcal{M}$, belonging to the same algorithm class of (10), be applied to the update (correction) step after function acquisition at t_{k+1}, for which one application on $\hat{\boldsymbol{x}}'_{k+1|k}$, delivers

$$\|\hat{\boldsymbol{x}}_{k+1} - \boldsymbol{x}^\star(t_{k+1})\| \leq \varrho_2 \|\hat{\boldsymbol{x}}'_{k+1|k} - \boldsymbol{x}^\star(t_{k+1})\|, \quad \varrho_2 \in (0,1) \tag{22}$$

for example, another proximal gradient step as

$$\hat{\boldsymbol{x}}_{k+1} = \operatorname{prox}_{\alpha g}\{\hat{\boldsymbol{x}}'_{k+1|k} - \alpha\nabla_{\boldsymbol{x}} f(\hat{\boldsymbol{x}}'_{k+1|k}; t_{k+1})\}. \tag{23}$$

Then the following result is in place.

Theorem 2 (Informal): Consider the time-varying Problem (15) and two methods $\mathcal{P}$ and $\mathcal{M}$ for which (20)–(22) hold. Let the predictor $\mathcal{P}$ be applied P times during the prediction step, and the corrector $\mathcal{M}$ be applied C times. Consider Assumption 2 to hold and additionally, let $f(\boldsymbol{x}; t)$ be L-smooth (in addition to be m-strongly convex), with a well-defined Hessian $\nabla_{\boldsymbol{x}\boldsymbol{x}} f(\boldsymbol{x}; t)$. Then, there exists a minimal number of prediction and correction steps P, C for which globally (i.e., starting from any initial condition)

$$\limsup_{k\to\infty} \|\hat{\boldsymbol{x}}_k - \boldsymbol{x}^\star(t_k)\| = \frac{\Delta_0}{m}\varrho_1^C\, O(h).$$

In addition, if we consider the assumption that higher-order derivatives of the cost function are bounded[6] as

$$\max\{\|\nabla_{\boldsymbol{x}\boldsymbol{x}\boldsymbol{x}} f(\boldsymbol{x}; t)\|, \|\nabla_{t\boldsymbol{x}\boldsymbol{x}} f(\boldsymbol{x}; t)\|, \|\nabla_{tt\boldsymbol{x}} f(\boldsymbol{x}; t)\|\} \leq \Delta_1$$

uniformly in time and for all $\boldsymbol{x} \in \mathbb{R}^n$, then locally (and for small h), there exists a minimal number of prediction and correction steps P, C so that

$$\limsup_{k\to\infty} \|\hat{\boldsymbol{x}}_k - \boldsymbol{x}^\star(t_k)\| = \underbrace{O(\Delta_1\varrho_1^C\, h^2)}_{\text{prediction gain}} + \underbrace{O(\Delta_0\varrho_1^C\varrho_2^P\, h)}_{\text{approximation error}}.$$

[6]Where induced Euclidean norms are considered for tensors.

Proof (Sketch): The proof here proceeds as follows: we first bound the error arising from the prediction [see (21)], and second bound the one from the correction [see (23)], and then finally combine them. For the error arising from the prediction, two errors must be considered, one arising from the model (due to the Taylor expansion error), whereas the other arising from the P prediction steps. When considering exact prediction ($P \to \infty$), the leading error is the Taylor expansion error [namely the error in (13)], which is $O(h)$, in general, and $O(h^2)$ when higher-order derivatives are bounded. ■

The results of Theorem 2 are fairly general and apply to different problem classes [27], [28]. Theorem 2 indicates that tracking is not worse than correction-only methods in the worst case. If the function has some higher degree of smoothness, and we are interested in a local result, then a better ATE can be achieved, provided some (stricter) conditions on the number of prediction and correction steps are verified. The ATE is composed of two terms: one which is labeled as approximation error, which is due to the early termination of the prediction step (if $P \to \infty$ and prediction is exact, this term goes to 0). The other, called the prediction gain, is the gain resulting from using a prediction step, which brings the error down to a $O(h^2)$ dependence on h. This depends on the first-order Taylor expansion employed; other methods can further reduce this to $O(h^4)$ or less [4], [29], [31]–[34] (look again at Fig. 2, where we have also employed a Taylor model up to degree 2 for (13) to obtain an $O(h^3)$ error bound).

The higher degree of smoothness required for the local results imposes boundedness of the tensor $\nabla_{\boldsymbol{xxx}} f(\boldsymbol{x}; t)$, which is a typical assumption for second-order algorithms (notice that the predictor requires second-order information, see (14)–(17) and its solution is comparable to solving a Newton step, which is locally quadratically converging). Moreover, it bounds the variability of the Hessian of f over time, which guarantees the possibility of performing more accurate predictions of the optimal trajectory. Theorem 2 depicts a key result in prediction–correction methods: the prediction value is fully exploited with higher smoothness.

D. Continuous-Time Algorithms

We consider now continuous-time prediction–correction algorithms which, in general, are appropriate in control and robotics applications.[7] The main component of these algorithms is the ability to track the minimizer by taking into account its evolution with time. In continuous-time, this scheme has been used in distributed time-varying convex optimization (see [5] and [39]–[41]). Since the objective function is m-strongly convex, the solution of the problem can be computed by solving the first-order optimality condition (12): for the implicit function theorem, the time derivative of $\boldsymbol{x}^\star(t)$ is

$$\dot{\boldsymbol{x}^\star}(t) = -\nabla^2_{\boldsymbol{xx}} f(\boldsymbol{x}; t)^{-1} \nabla_{t\boldsymbol{x}} f(\boldsymbol{x}; t). \tag{24}$$

In cases where the problem of interest is static, gradient descent and Newton's method can be used, for instance, to find trajectories such that $\lim_{t\to\infty} \boldsymbol{x}(t) = \boldsymbol{x}^\star$. Moreover, this convergence is exponential, meaning that there exist positive constants C_1 and α_1 such that $\|\boldsymbol{x}(t) - \boldsymbol{x}^\star\| \leq C_1\, e^{-\alpha_1\, t}$ (see [42, Definition 4.5]—note that exponential convergence is the continuous counterpart of the discrete-time Q-linear convergence). To provide the same guarantees in the case of time-varying optimization, we include the prediction term (24), which incorporates changes in the optimizer

$$\dot{\boldsymbol{x}}(t) = -\nabla^2_{\boldsymbol{xx}} f(\boldsymbol{x}; t)^{-1} \left(\kappa \nabla_{\boldsymbol{x}} f(\boldsymbol{x}; t) + \nabla_{t\boldsymbol{x}} f(\boldsymbol{x}; t)\right) \tag{25}$$

where $\kappa > 0$ is referred to as "gain of the controller" in the literature, and (25) is referred to as "the controller," since it controls how the decision trajectory must change to reach the optimal solution trajectory. This differential equation defines a nonautonomous dynamical system which converges exponentially to $\boldsymbol{x}^\star(t)$ [43], [6, Prop. 1].

Theorem 3: Under the hypothesis of Theorem 2, $\boldsymbol{x}(t)$—the solution of the dynamical system (25)—converges exponentially to $\boldsymbol{x}^\star(t)$, solution to (1).

Proof (Sketch): The proof uses a Lyapunov argument. Define the error $\boldsymbol{e}(t) := \boldsymbol{x}(t) - \boldsymbol{x}^\star(t)$ and the function $V(\boldsymbol{e}; t) = \|\nabla_{\boldsymbol{x}} f(\boldsymbol{e} + \boldsymbol{x}^\star(t); t)\|^2 / 2$. Then the proof relies on establishing that $\dot{V}(\boldsymbol{e}; t) < 0$ for all $\boldsymbol{e} \neq 0$ and $\dot{V}(\boldsymbol{0}; t) = 0$ (see [42, Th. 4.10]), and in particular

$$\dot{V}(\boldsymbol{e}; t) = -\kappa \|\nabla_{\boldsymbol{x}} f(\boldsymbol{e} + \boldsymbol{x}^\star(t); t)\|^2 \leq 0.$$

■

This result indicates that the convergence is exponential to the optimal trajectory (ATE is zero). The latter is achieved by including the prediction in the controller, that is, the time variation of the optimal solution. Without such a predictor, tracking would be possible only up to an asymptotic error that depends on the variation of the gradient with the time and the gain of the controller. This is a clear benefit of structured algorithms. Note that these results are the continuous-time counterpart of the results presented in Theorem 2. However, one of the advantages of working with continuous-time flows is that it is also possible to establish asymptotic convergence to the solution of constrained optimization problems using interior point

[7] For these algorithms, time metrics like TR make less sense than in discrete-time setting. However, continuous-time algorithms are still interesting to investigate, both in theory—as continuous limits to discrete-time algorithms—and in practice, as good approximation of cases in which the sampling time is much smaller than other system characteristic times.

methods (see [14, Chapter 11]). Formally, let us define the following optimization problem:

$$\boldsymbol{x}^\star(t) := \operatorname*{argmin}_{\boldsymbol{x}\in\mathbb{R}} \quad f(\boldsymbol{x};t) \tag{26a}$$

$$\text{s.t. } h_i(\boldsymbol{x};t) \leq 0 \quad \forall i = 1,\dots,p. \tag{26b}$$

In [6], inspired by interior point methods, the following barrier function is proposed:

$$\Phi(\boldsymbol{x};t) = f(\boldsymbol{x};t) - \frac{1}{c(t)}\sum_{i=1}^{p}\log\left(s(t) - h_i(\boldsymbol{x};t)\right) \tag{27}$$

where $c(t)$ is an increasing function such that $\lim_{t\to\infty} c(t) = \infty$ and $s(t) = s(0)e^{-\gamma t}$ for some $\gamma > 0$. The intuition behind the barrier is that it approximates the indicator function as t increases. This means that it takes the value 0 when the constraint is satisfied and $+\infty$ in the opposite case. In that sense, when minimizing the unconstrained objective $\Phi(\boldsymbol{x};t)$ constraint satisfaction is promoted. Note that for the logarithm to be well defined we need $s(t) > h_i(\boldsymbol{x};t)$ and thus the slack $s(t)$ is introduced just to guarantee that this is the case at all times $t \geq 0$. In particular, it suffices to choose $s(0) \geq \max_{i=1,\dots,p}\{h_i(\boldsymbol{x}(0),0)\}$ for this to be the case [6, Th. 1]. The previous intuition on how minimizing the function $\Phi(\boldsymbol{x};t)$ defined in (27) resembles to solve (26) can be formally established. Let $\hat{\boldsymbol{x}}(t)$ be the minimizer of $\Phi(\boldsymbol{x};t)$. Then it follows that $\lim_{t\to\infty}\|f(\hat{\boldsymbol{x}}(t);t) - f(\boldsymbol{x}^\star(t);t)\| = 0$ [6, Lemma 1]. This result, along with the idea that the barrier function promotes constraint satisfaction, suggests that to solve (26), it suffices to compute the minimizer of the unconstrained barrier function $\Phi(\boldsymbol{x};t)$ defined in (27). This result is formalized in the following theorem.

Theorem 4 (Theorem 1 [6]): Consider the constrained optimization problem defined in (26) with $f(\boldsymbol{x};t)$ m-strongly convex, $h_i(\boldsymbol{x};t)$ for all $i = 1,\dots,p$ are convex functions and Slater's constraint qualifications hold: that is, there exists $\boldsymbol{x}^\dagger(t)$ such that for all $t \geq 0$ and for all $i = 1,\dots,p$, it holds that $h_i(\boldsymbol{x}^\dagger(t),t) < 0$. Let $\Phi(\boldsymbol{x};t)$ be the barrier defined in (27) and let $\boldsymbol{x}(t)$ be the solution of the dynamical system

$$\dot{\boldsymbol{x}}(t) = -\nabla_{\boldsymbol{x}\boldsymbol{x}}\Phi(\boldsymbol{x};t)^{-1}\left(\kappa\nabla_{\boldsymbol{x}}\Phi(\boldsymbol{x};t) + \nabla_{t\boldsymbol{x}}\Phi(\boldsymbol{x};t)\right).$$

Then it follows that $\lim_{t\to\infty}\|\boldsymbol{x}(t) - \boldsymbol{x}^\star(t)\| = 0$.

Proof: The proof follows that of Theorem 3 with $\boldsymbol{e} := \boldsymbol{x} - \boldsymbol{x}^\star(t)$ and Lyapunov function $V(\boldsymbol{e};t) = \|\nabla_{\boldsymbol{x}}\Phi(\boldsymbol{e} + \boldsymbol{x}^\star(t);t)\|^2/2$. ■

Working in continuous time allows us to solve constrained problems using interior point methods, thus guaranteeing feasibility for all time if the initial solution is feasible. This is especially appropriate for control systems where the constraints might represent physical constraints that need to be satisfied for the system to operate without failure.

III. APPLICATIONS

We highlight now application domains where structured and unstructured time-varying optimization methods have been or could be applied to. We proceed with a high-level (and by no means exhaustive) list of areas, presented in alphabetical order. Note that, given the increasingly cross-disciplinary nature of the research efforts, clear boundaries are difficult to delineate.

A. Communications

Problems such as congestion control, resource allocation, and power control have been of paramount importance in communication networks [44], [45]. Indeed, important questions arise when channel capacities and noncontrollable traffic flows are time-varying, with changes that are faster than the solution time of underlying optimization tasks, and even more so in the 5G era [46] (e.g., HD video streaming). This setting can be tackled with time-varying optimization tools. For example, in [47], a continuous-time structured algorithm with a first-order Taylor predictor model is proposed. The recent work [20] explored structured algorithms for intermittent time-varying service, a feature important in today's cloud computing. Finally, time variations are important when the communication graph is itself time-varying (see [48] and references therein).

B. Control Systems

One popular tool in control systems is MPC [49]. MPC is grounded on a strategy where an optimization problem is formulated to compute optimal states and commands for a dynamical system over a given temporal window; once a solution is identified, the command for the first time instant is implemented and the window is then shifted. The optimization problem changes over time, since it is parametrized over the state of a certain system, and it has to be resolved every time. Recently, time-varying (and/or parameter-varying) algorithms for MPC have appeared for large-scale and embedded systems (see [13] and [50]–[52]), which are a mix of continuous-time and discrete-time unstructured and structured algorithms. For example, in [51], an unstructured algorithm (specifically a homotopy-based continuation method) is used to enhance the tracking performance of the nonlinear MPC. In [13], a predictor of the form (14) is used to solve the optimization problem that arises from the receding horizon problem. Since the solution varies smoothly with the state of the system, these methods are appropriate to achieve good tracking accuracy with low computational cost. In [53], these ideas are extended to problems with constraints by using a semismooth barrier function.

Other applications in control systems are the sequential training of neural networks for online system identification [43], [54], [55], where predictors of the form (25) were proposed, as well as recent work at the intersection of online optimization and feedback control, where

the output regulation problem is revisited by posing the problem of driving the output of a dynamical system to the optimal solution of a time-varying optimization problem [56], [57].

C. Cyber–Physical Systems

Cyber–physical systems (CPSs) [58] are engineered systems with tightly integrated computing, communication, and control technologies. Because of major technological advances, existing CPSs (power systems, transportation networks, and smart cities just to mention a few) are evolving toward societal-scale systems operating in highly dynamic environments, and with a massive number of interacting entities. It is then imperative to revisit information processing and optimization tools to enable optimal and reliable decision-making on time scales that match the dynamics of the underlying physical systems. Due to space limitations, we focus here on power systems and transportation systems.

A time-varying problem for power systems can capture variations at a second level in noncontrollable loads and available power from renewables [59]; it can also accommodate dynamic pricing schemes. Time-varying problem formulations (and related online algorithms) can be utilized for tasks such as demand response, optimal power flow (OPF), and state estimation. Adopting a time-varying optimization strategy, the power outputs of DERs can be controlled at the second level to regulate voltages and currents within limits in the face of volatility of renewables and noncontrollable loads and to continually steer the network operation toward points that are optimal based on the formulated time-varying problem. Examples of works include real-time algorithms for voltage control, OPF, as well as DER management for aggregators (see, for example, [60]–[65] and pertinent references therein). For some applications, such as the demand response and the OPF, online algorithms have been designed to leverage measurements of constraints (e.g., voltages violations) in the algorithmic updates [18], [61], [66] to relax the sensing requirements. Real-time measurements were used in a state estimation framework in [67]. We develop these ideas with an example in Section IV-A.

In the context of transportation systems, fast time variations may arise from different factors (and at appropriate time scales), such as variations in the traffic, pedestrians crossing the roads, car accidents, sport events; these factors may lead to time-dependent routing and traffic light control algorithms [68]. Motivated by the recent widespread use of ridesharing and mobility-on-demand services [69], spatio-temporal variations naturally emerge from customer pick-up and drop-off requests as well as fleet locations. As representative works in context, Alonso-Mora *et al.* [69] and Simonetto *et al.* [70] discussed unstructured algorithms to achieve long-term ("asymptotical") good tracking, while sacrificing short-term optimality. In [71], an online algorithm based on a structured problem formulation is presented, where the prediction is based on historical data and machine learning forecasting. An unstructured algorithm is also presented in [72], to find optimal meeting points.

D. Machine Learning and Signal Processing

As a representative problem spanning the broad fields of machine learning and signal processing, we focus on the reconstruction of sparse signals via ℓ_1-regularization where we are interested in recovering a sparse signal given some observations, for example, extract "sparse" features in images [73]. The time-varying nature of this problem arises when we want, for instance, to extract features in videos. Works that explore dynamic ℓ_1 reconstruction are, for example, [74]–[76]. In [35] and [77], two algorithms are presented, one unstructured using homotopy and one structured building a model based on methods akin to Kalman filters. In [78], unstructured methods for the elastic net are discussed.

Other applications in machine learning and signal processing, where a number of (mainly) unstructured algorithms have been proposed, include contemporary approaches for sparse, kernel-based, robust, linear regression, zeroth-order methods, and learning problems over networks. Additional lines of work include dynamic classification under concept drift [79], dynamic beamforming [80], and other dynamic signal processing tasks, such as maximum *a posteriori* estimation [81], [82].

E. Medical Engineering

Medical engineering is a growing research field in many contexts. Here, we focus on the new possibilities offered by new and fast imaging modalities under magnetic resonance (see [83] and [84] for a broader context). Once confined to static images (due to the high computational load), magnetic resonance imaging (MRI) is now transitioning to fast imaging and possibly high-definition video streaming, which could be of invaluable help to clinicians and researchers alike, not to mention patients, especially children. In the series of work [85] and [86], the authors describe an unstructured algorithm to solve a time-varying subsampled nonlinear regularized inverse problem. The algorithm allows the clinicians to visualize blood flow, cardiac features, and swallowing, among many other things.

F. Optimization and Mathematical Programming

Time-varying optimization has been studied for applications within mathematical programming, for example, in the context of parametric programming [3], [4], [7], [37], [87] where a wealth of structured and unstructured algorithms are presented. Time-varying optimization has its roots in continuation methods in numerical mathematics [38] and it resembles path-following methods [88], so advances in either fields are intertwined.

Another application in mathematical programming where time-varying optimization could be (and has been)

applied is the field of evolutionary variational inequalities. Variational inequalities [89] can be framed as optimization problems, while evolutionary ones can be framed as time-varying optimization problems. In [90]–[92], the authors discuss plenty of interesting applications in socio-economical sciences (human migration studies, economics, time-dependent equilibria in games, etc.), proposing mainly unstructured approaches.

G. Process Engineering

In chemical and process engineering, the body of work [93]–[95] focuses on real-time optimization for chemical and industrial processes. The optimization problem is not time-varying per se, but it becomes time-varying because the constraints (i.e., the industrial process) are learned online and adapted. Several real-time optimization algorithms are proposed, mainly unstructured.

H. Robotics

Time-varying optimization problems—or problems that depend on a time-varying parameter—appear often in the context of robotic systems. In the context of safe navigation, Arslan and Koditschek [96], [97] considered the problem of using power diagrams to define a local safe space, which depends on the position of the agent itself. The control law used to navigate is such that it aims to track the projection of the goal on the local safe space. Even in cases where the goal is static, a time-varying optimization problem needs to be solved due to the modification of the local free space. In [6], the approach described in Section II-D is used to compute said solutions. We develop these ideas more in Section IV-B.

For networks of mobile robots [98], the "communication integrity" is guaranteed by solving a time-varying optimization problem. Specifically, since an unstructured algorithm is used, an asymptotic tracking error that results in small constraint violation and suboptimality is achieved.

Another interesting application is that of robotic manipulators [32], [99], [100], obtained via zeroing neural dynamics (ZND) [101]–[103], based on a prediction step similar to (25).

IV. TWO APPLICATIONS: DEEP DIVE

A. Example in Power Grids

Consider a power distribution grid serving residential houses or commercial facilities, featuring N controllable DERs. The vector $\boldsymbol{x}_i \in X_i \subset \mathbb{R}^2$ collects the active and reactive power outputs of the ith DER, and X_i models hardware constraints. A prototypical time-varying optimization problem for real-time management of DERs is

$$\boldsymbol{x}^\star(t_k) := \operatorname*{argmin}_{\{\boldsymbol{x}_i \in X_i\}_{i=1}^N} \sum_{i=1}^{N} f_i(\boldsymbol{x}_i; t_k) + f_{N+1}(\boldsymbol{x}; t_k) \quad (28)$$

where $f_i(\boldsymbol{x}_i; t_k)$ is a cost function associated with the ith DER and $f_{N+1}(\boldsymbol{x}; t_k)$ is a time-varying cost associated with the power network operator. Elaborating on the latter, suppose, for example, that a linearized model for the power flow equations is utilized to capture the variations on some electrical quantities $\boldsymbol{y} \in \mathbb{R}^m$ (e.g., voltages and power flows on lines) induced by $\boldsymbol{x}$, that is, $\boldsymbol{y}(t_k) = \boldsymbol{A}_x \boldsymbol{x} + \boldsymbol{A}_w \boldsymbol{w}(t_k)$, where $\boldsymbol{w}(t_k)$ is a vector collecting the powers of noncontrollable devices and $\boldsymbol{A}_x, \boldsymbol{A}_w$ are sensitivity matrices that are built based on the network topology and the line impedances [61], [65]. A possible choice for the function $f_{N+1}(\boldsymbol{x}; t_k)$ for the network operator can then be $f_{N+1}(\boldsymbol{x}; t_k) = \frac{\gamma}{2}\|\boldsymbol{y}^{\text{ref}}(t_k) - \boldsymbol{A}_x \boldsymbol{x} + \boldsymbol{A}_w \boldsymbol{w}(t_k)\|^2$, where $\boldsymbol{y}^{\text{ref}}(t_k)$ is a time-varying reference point for the electrical quantities included in $\boldsymbol{y}$, and $\gamma > 0$ is a design parameter that influences the ability to track the time-varying reference signal $\boldsymbol{y}^{\text{ref}}(t_k)$. Various models for $f_i(\boldsymbol{x}_i; t_k)$ can be adopted, based on specific problem settings, for example, $f_i(\boldsymbol{x}_i; t_k) = \|\boldsymbol{x}_i - \boldsymbol{x}_i^{\text{ref}}(t_k)\|^2$ can minimize the deviation from a desirable setpoint for the ith DER (that can be computed based on a slower time-scale dispatch problem); in the case of photovoltaic systems, $\boldsymbol{x}_i^{\text{ref}}(t_k)$ could be set to $\boldsymbol{x}_i^{\text{ref}}(t_k) = [P^{\text{av}}(t_k), 0]^T$, with $P^{\text{av}}(t_k)$ the maximum power available, to minimize the power curtailed. Alternatively, set $f_i(\boldsymbol{x}_i; t_k)$ to a time-varying incentive $-\boldsymbol{\pi}_i^T(t_k)\boldsymbol{x}_i$ to maximize the profit of the ith DER in providing services to the grid.

With reference to Fig. 1, in this application, data streams include the parameters of the time-varying function $f_i(\boldsymbol{x}_i; t_k)$ (e.g., the power setpoints $\{\boldsymbol{x}_i^{\text{ref}}(t_k)\}$ or the incentive signals $\{\boldsymbol{\pi}_i(t_k)\}$), the function $f_{N+1}(\boldsymbol{x}; t_k)$ (where set points $\boldsymbol{y}^{\text{ref}}(t_k)$ can rapidly change to provide various services to the grid), as well as the powers $\boldsymbol{w}(t_k)$ consumed by the noncontrollable devices. The algorithm produces decisions on setpoints for the active and reactive power outputs $\boldsymbol{x}_i(t_k)$ of the DERs, which are commanded to the devices. Finally, "feedback" can come in the form of measurements of the actual power outputs $\boldsymbol{x}_i(t_k)$ [60], as well as other electrical quantities [61], [66].

As an illustrative example, we consider the case where $N = 500$ DERs are controlled in a distribution feeder; the set X_i is built so that the ranges of active and reactive powers are $[-50, 50]$ kW and $[-50, 50]$ kVAr, and $f_i(\boldsymbol{x}_i; t_k)$ is set to $f_i(\boldsymbol{x}_i; t_k) = \frac{1}{2}\|\boldsymbol{x}_i\|^2$ for all DERs. This setting is representative of a case where energy storage resources are utilized to provide services. We consider the case where $\boldsymbol{y}$ is a scalar and represents the net power consumed by a distribution network; in this case, $\boldsymbol{y}^{\text{ref}}(t_k)$ can model automatic generation control (AGC) signals or flexible ramping signals. The matrices $\boldsymbol{A}_x$ and $\boldsymbol{A}_w$ are built as in [61]. We use the real data provided in [61] to generate the vectors $\boldsymbol{w}(t_k)$ with a granularity of 1 s. The parameters are $m = 1$, $L = 21$, and $\gamma = 2$; the step size is $\alpha = 1/(10\,L)$. We keep the computational time fixed in our comparison between the unstructured running projected gradient and the structured prediction–correction algorithm; in particular, we consider the cases $P = 0, C = 3$, and $P = 3, C = 1$ (see Theorem 2).

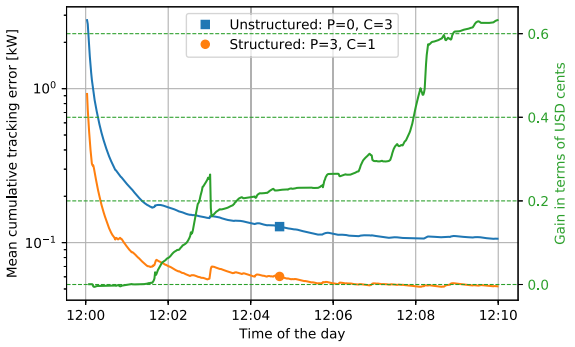

Fig. 5. *Mean cumulative tracking error $(1/T)\sum_{k=1}^{T}\|x(t_k)-x^\star(t_k)\|$ versus time of the day for a choice of structured ($P=3$, $C=1$) and unstructured ($C=3$) algorithms, having the same computational time. In green, we report the hypothetical gain in terms of less utilized power at the average cost of 12 USD cent per kWh.*

To outline the steps of the prediction–correction algorithm, recall that $\hat{\boldsymbol{x}}_k$ denotes the iterate of the algorithm at time t_k [see Th. 1], and let $f(\boldsymbol{x};t_k) := \sum_{i=1}^{N} f_i(\boldsymbol{x}_i;t_k) + f_{N+1}(\boldsymbol{x};t_k)$ and $X = X_1 \times X_2 \times \cdots \times X_N$ for brevity. A prediction $\hat{\boldsymbol{x}}_{k|k-1}$ is obtained by running P prediction steps $p = 0,\dots,P-1$:

$$\hat{\boldsymbol{x}}^{p+1} = \text{proj}_X\{\hat{\boldsymbol{x}}^p - \alpha(\nabla_{\boldsymbol{xx}} f(\hat{\boldsymbol{x}}_k;t_k)(\hat{\boldsymbol{x}}^p - \hat{\boldsymbol{x}}_{k-1}) + h\nabla_{t\boldsymbol{x}} f(\hat{\boldsymbol{x}}_{k-1};t_k) + \nabla_{\boldsymbol{x}} f(\hat{\boldsymbol{x}}_{k-1};t_k))\} \quad (29)$$

and by setting $\hat{\boldsymbol{x}}_{k|k-1} = \hat{\boldsymbol{x}}^P$. Starting now from $\bar{\boldsymbol{x}}^0 = \hat{\boldsymbol{x}}_{k|k-1}$, the correction phase involves the following C steps:

$$\bar{\boldsymbol{x}}^{c+1} = \text{proj}_X\{\hat{\boldsymbol{x}}^p - \alpha(\nabla_{\boldsymbol{x}} f(\bar{\boldsymbol{x}}^c;t_k))\} \quad (30)$$

for $c = 0,1,\dots,C-1$. The iterate $\hat{\boldsymbol{x}}_k$ is then $\hat{\boldsymbol{x}}_k = \bar{\boldsymbol{x}}^C$. Note that if $P = 0$, one recovers the unstructured running projected gradient method [see also (9)]. In the simulations, the time derivative $\nabla_{t\boldsymbol{x}} f(\hat{\boldsymbol{x}}_k;t_k)$ in (29) is substituted by an approximate version (see, e.g., [27] and [104]).

To assess the performance of the prediction–correction algorithm, Fig. 5 depicts the mean cumulative tracking error $(1/T)\sum_{k=1}^{T}\|\boldsymbol{x}(t_k) - \boldsymbol{x}^\star(t_k)\|$. It can be seen that by leveraging the temporal structure of the problem, the prediction–correction algorithm offers improved performance. We can now evaluate the performance metrics presented in Section II. We compute the ATE as the mean error in the last 20 s of the simulation, yielding an ATE of ∼50 W for the unstructured method, and an ATE of ∼80 W for the structured method. Since the computational time of both methods is the same, it follows that SG = 1.6. The CR can be empirically evaluated by the time it takes to enter the ATE ball as approximately 1 min for both methods. On the other hand, the TR is hardware-dependent, since the denominator of the TR depends on the computational capabilities of the microcontrollers embedded in the DERs, where algorithms are implemented.

B. Example in Robotics

Consider a navigation setup of driving a disk-shaped robot of radius $r > 0$, whose position is denoted by $\boldsymbol{x}_\mathsf{r}(t)$, to a desired configuration $\boldsymbol{x}_\mathsf{d}(t)$, while avoiding collisions with obstacles in the environment. Here, we deal with a closed and convex workspace $\mathcal{W} \subset \mathbb{R}^n$ of possible configurations that the robot can take. Assume that the workspace is populated with m nonintersecting spherical obstacles, where the center and radius of the ith obstacle are denoted by $\boldsymbol{x}_i \in \mathcal{W}$ and $r_i > 0$, respectively. In general, this navigation problem is nonconvex due to the presence of obstacles; however, one can convexify it by looking at the collision-free convex local workspace around $\boldsymbol{x}_\mathsf{r}$, defined as [96]

$$\mathcal{LF}(\boldsymbol{x}_\mathsf{r}) = \left\{\boldsymbol{x} \in \mathcal{W}\colon (\boldsymbol{x}_i - \boldsymbol{x}_\mathsf{r})^\top \boldsymbol{x} - b_i(\boldsymbol{x}_\mathsf{r}) \le 0,\ i = 1\dots m\right\}$$

where $b_i(\boldsymbol{x}_\mathsf{r})$ are pertinent scalars computed depending on robot and obstacles positions (see [96]). The collision-free local workspace describes a local neighborhood of the robot that is guaranteed to be free of obstacles. Each obstacle introduces a linear bound and thus the local free space is convex and yields a polygon as the blue colored one in Fig. 6 (see [96, Eq. (6)]). The position of the target $\boldsymbol{x}_\mathsf{d}(t)$, the location of the robot $\boldsymbol{x}_\mathsf{r}(t)$, and the local free space $\mathcal{LF}(\boldsymbol{x}_\mathsf{r})$ correspond to the data stream of Fig. 1. Supposing that the robot follows the integrator dynamics $\dot{\boldsymbol{x}}_\mathsf{r} = \mathbf{u}(\boldsymbol{x}_\mathsf{r})$, the controller proposed in [96] is given by $\dot{\boldsymbol{x}}_\mathsf{r}(t) = -G_\mathsf{c}(\boldsymbol{x}_\mathsf{r} - \boldsymbol{x}^\star(t))$, where $G_\mathsf{c} > 0$ and $\boldsymbol{x}^\star(t)$ are the orthogonal projections of the desired configuration $\boldsymbol{x}_\mathsf{d}(t)$ onto the collision-free local workspace $\mathcal{LF}(\boldsymbol{x}_\mathsf{r})$. Since the local workspace is collision-free, so is the direction $\boldsymbol{x}_\mathsf{r} - \boldsymbol{x}^\star(t)$, and thus the control law is guaranteed to avoid the obstacles. This controller also guarantees that the robot converges to $\boldsymbol{x}_\mathsf{d}(t)$ [96]. It requires computing the projection of $\boldsymbol{x}_\mathsf{d}(t)$ onto $\mathcal{LF}(\boldsymbol{x}_\mathsf{r})$ by solving the time-varying convex problem

$$\boldsymbol{x}^\star(t) := \underset{\boldsymbol{x}\in\mathcal{LF}(\boldsymbol{x}_\mathsf{r})\subseteq\mathbb{R}^n}{\text{argmin}} \ \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{x}_\mathsf{d}(t)\|^2. \quad (31)$$

By using the barrier function defined in (27) and the dynamics in Theorem 4, one can compute $\hat{\boldsymbol{x}}(t)$, an estimate of $\boldsymbol{x}^\star(t)$ and apply the control law $\dot{\boldsymbol{x}}_\mathsf{r}(t) = -G_\mathsf{c}(\boldsymbol{x}_\mathsf{r} - \hat{\boldsymbol{x}}(t))$. The barrier function in (27) for this application takes the form

$$\Phi(\boldsymbol{x},\boldsymbol{x}_\mathsf{r};t) = \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{x}_\mathsf{d}(t)\|^2 - \frac{1}{c(t)}\sum_{i=1}^{m}\log(b_i(\boldsymbol{x}_\mathsf{r}) - \mathbf{a}_i(\boldsymbol{x}_\mathsf{r})^\top \boldsymbol{x})$$

with $\mathbf{a}_i = \boldsymbol{x}_i - \boldsymbol{x}_\mathsf{r}$. Then estimate $\hat{\boldsymbol{x}}(t)$ is the solution of the following dynamical system with initial condition

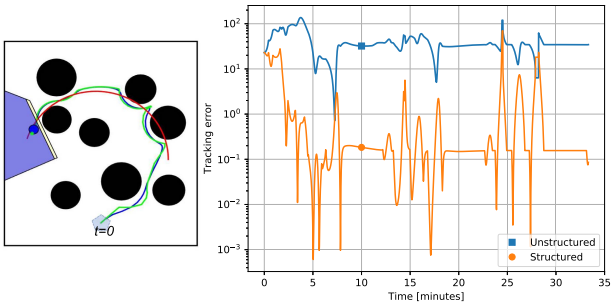

Fig. 6. *Left: The red circle represents the desired $x_d(t)$. The green and blue lines represent, respectively, the trajectories of the estimates of the projected goal $\hat{x}(t)$ and the trajectories of the robot $x_r(t)$ for the structured algorithm. Right: Tracking error $\|x(t_k) - x^*(t_k)\|$ versus time for a choice of structured and unstructured algorithm.*

$\hat{\boldsymbol{x}}(0) = \boldsymbol{x}_{\mathrm{r}}(0)$:

$$\dot{\hat{\boldsymbol{x}}}(t) = -\nabla_{\boldsymbol{x}\boldsymbol{x}}\Phi(\boldsymbol{x}, \boldsymbol{x}_{\mathrm{r}}; t)^{-1}\left(\kappa\nabla_{\boldsymbol{x}}\Phi(\boldsymbol{x}, \boldsymbol{x}_{\mathrm{r}}; t) + \nabla_{\boldsymbol{x}t}\Phi(\boldsymbol{x}, \boldsymbol{x}_{\mathrm{r}}; t)\right)$$

where $c(t) = 1\ e^{0.1\ t}$ and $\kappa = 0.1$. To evaluate the performance of the proposed controller and optimizer, we consider a workspace $\mathcal{W} = [-20, 20] \times [-20, 25]$ containing eight circular obstacles (black circles in Fig. 6-left). Fig. 6(left) also depicts the trajectories followed by a disk-shaped robot of radius equal to 1 (blue circle) where $G_{\mathrm{c}} = 2$. The red line represents the trajectory of $\boldsymbol{x}_{\mathrm{d}}(t)$ and the green and blue lines represent, respectively, the trajectories of the estimates $\hat{\boldsymbol{x}}(t)$ of the projected goal onto the collision-free local workspace, and the trajectories of the center of mass of the robot $\boldsymbol{x}_{\mathrm{r}}(t)$.

In Fig. 6(right), we plot the metric defined in (3) for the algorithm with and without prediction, that is, structured and unstructured, respectively. Evidently, there is significant benefit using the structured algorithm.

V. RESEARCH OUTLOOK AND FUTURE CHALLENGES

Time-varying optimization is rapidly arising as an attractive algorithmic framework for today's fast-changing complex systems and world-size networks that entail heterogeneous and spatially distributed data streams. This article delineated the framework and underlined that structured algorithms can offer improved solutions to time-varying problems. In this section, a brief and certainly nonexhaustive list of the current challenges for structured and unstructured methods is outlined, with due implications in a number of potential applications.

A. Wider Classes of Problems

It has already been argued that unstructured methods generally require less functional assumptions than structured ones. For example, unstructured methods have been proposed for various nonstrongly convex problems, as well nonconvex cost functions, where notions of dynamic regret can be used as performance indicators (see [17], [18], [65], and [105]–[107]). An attractive feature of time-varying nonconvex optimization algorithms is that they can be free of locally optimal trajectories. For structured methods, these classes of problems are largely unexplored, since, for example, underlying evolution models will have to be set-valued for nonstrictly convex time-varying problems (because the solution trajectory is not unique). Interesting questions regarding bifurcations and merging of locally optimal trajectories, as well as the possibility of escaping isolated locally optimal trajectories naturally arise in this setting. A few efforts in this direction are included in [3], [4], and [38], but a comprehensive framework is lacking. A possible venue in this area could rely on piecewise linear continuation methods [38].

B. Data-Driven Models

Dynamic means of capturing the underlying optimization trajectory are now largely based on models, while in the current data streaming era, problems are often constructed in a data-driven fashion (e.g., via zero-order/bandit methods [108] or in a Bayesian setting [109]). Constructing and learning dynamic models for the optimization trajectory (e.g., via historical data) is a largely unexplored territory, especially for structured methods, where high-order smoothness is required for enhanced performance, in contrast with what typically (noisy) zero-order methods can provide. Unstructured methods can be found in [110]–[112].

C. Distributed Architectures

Distributed methods to solve time-varying optimization problems (possibly involving large-scale networks) are key in many contemporary cyber-physical applications. Both structured and unstructured methods have been investigated [5], [17], [39]–[41], [48], [80]–[82], [98], [104], [113], but many challenges remain. As discussed in [114], most distributed methods rely on diminishing step-size rules, which might not be an appropriate choice in time-varying settings when the algorithm runs indefinitely (as in, for example, video surveillance and monitoring of critical infrastructure). Another insight from [113], [114], and [115] is that the convergence behavior of distributed algorithms in the online setting is different relative to the batch case: traditional hierarchies in terms of convergence may be "flipped," with the slowest algorithm in the static case being the fastest algorithm in the time-varying one. In addition, the notion of asynchronous updates assumes a more prominent position, inasmuch the network of computing nodes may have access to different evolution models, sample the optimization problem at different time steps, at different time scales, or deliver solutions with different accuracy. All of this hinders standard analysis and it remains largely unexplored.

D. Feedback Loop

As we have seen in the analytical results presented here, under the assumptions provided, the time-varying algorithms converge to an error bound. Two key aspects are that: 1) the error bound can be arbitrarily big, if the algorithm converges arbitrarily slow, that is, if ϱ is arbitrarily close to 1 and 2) the time-varying algorithms are considered separately, meaning the decision stream $\hat{\boldsymbol{x}}(t)$ does not influence the optimization problem at future times. Ensuring "close-loop" stability and performance, when the decision stream is fed back to the system is a mostly open challenge, and one can expect that arbitrarily slow algorithms cause lack of convergence. In this case, the very notion of ATE may be ill-defined or too hard to achieve, since typically the cost will be parametrized also on the approximated optimizer trajectory, and system-oriented notions of stability and robustness may be more appropriate. Some initial work can be found in [13], [51], and [53] in the context of MPC, yet this area remains largely open.

Another emerging research topic is the development of online structured and unstructured online algorithms that effectively act as feedback controlled dynamical systems. The main goal is to drive the output of a dynamical system to solutions of time-varying optimization problems. Initial efforts toward unstructured online algorithms include [56], [57], where a Lyapunov analysis is also provided, and the more recent works in [116] and [117] that provide a pertinent regret analysis).

E. Interactive and Reinforcement Learning (RL)

We close with potential links of the time-varying optimization tools outlined in this article with related contemporary thrusts on OCO [9], bandits [118], and RL that encompasses interactive decision making between agents and generally dynamic environments [119]. At this stage, these links are active research thrusts that are pursued in diverse applications, such as allocation of network resources, secure mobile edge computing, and management of Internet-of-Things (see [120] and [121], and references therein). Clearly, at the core of OCO, bandits, and RL are sequential solutions of optimization objectives that vary as the environment transitions across states and the agents take actions dynamically. These key elements prompt one to foresee that the time-varying tools overviewed in the present article can be fruitfully leveraged in interactive optimization. One key challenge to bear in mind in this direction is that the objective function in RL changes not only due to time-varying effects, but also due to actions fed back by the agent (learner). How to broaden the scope of algorithms presented here in such a wider context constitutes an exciting open research direction. ■

REFERENCES

[1] J. J. Moreau, "Evolution problem associated with a moving convex set in a Hilbert space," *J. Differ. Equ.*, vol. 26, no. 3, pp. 347–374, Dec. 1977.

[2] B. T. Polyak, *Introduction to Optimization*. New York, NY, USA: Optimization Software, 1987.

[3] J. Guddat, F. G. Vazquez, and H. T. Jongen, *Parametric Optimization: Singularities, Pathfollowing Jumps*. Chichester, U.K.: Wiley, 1990.

[4] 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 J. Control Optim.*, vol. 51, no. 3, pp. 1823–1840, Jan. 2013.

[5] S. Rahili and W. Ren, "Distributed convex optimization for continuous-time dynamics with time-varying cost functions," *IEEE Trans. Autom. Control*, vol. 62, no. 4, pp. 1590–1605, Apr. 2017.

[6] M. Fazlyab, S. Paternain, V. M. Preciado, and A. Ribeiro, "Prediction-correction interior-point method for time-varying convex optimization," *IEEE Trans. Autom. Control*, vol. 63, no. 7, pp. 1973–1986, Jul. 2018.

[7] V. M. Zavala and M. Anitescu, "Real-time nonlinear optimization as a generalized equation," *SIAM J. Control Optim.*, vol. 48, no. 8, pp. 5444–5467, Jan. 2010.

[8] A. Simonetto and E. Dall'Anese, "Prediction-correction algorithms for time-varying constrained optimization," *IEEE Trans. Signal Process.*, vol. 65, no. 20, pp. 5481–5494, Oct. 2017.

[9] S. Shalev-Shwartz, "Online learning and online convex optimization," *Found. Trends Mach. Learn.*, vol. 4, no. 2, pp. 107–194, 2011.

[10] R. Dixit, A. S. Bedi, R. Tripathi, and K. Rajawat, "Online learning with inexact proximal online gradient descent algorithms," *IEEE Trans. Signal Process.*, vol. 67, no. 5, pp. 1338–1352, Mar. 2019.

[11] N. Bastianello, A. Simonetto, and R. Carli, "Prediction-correction splittings for nonsmooth time-varying optimization," in *Proc. Eur. Control Conf.*, Napoli, Italy, Jun. 2019, pp. 1963–1968.

[12] G. A. Hicks and W. H. Ray, "Approximation methods for optimal control synthesis," *Can. J. Chem. Eng.*, vol. 49, no. 4, pp. 522–528, Aug. 1971.

[13] S. Paternain, M. Morari, and A. Ribeiro, "A prediction-correction algorithm for real-time model predictive control," 2019, *arXiv:1911.10051*. [Online]. Available: http://arxiv.org/abs/1911.10051

[14] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[15] R. Rockafellar, *Convex Analysis*. Princeton, NJ, USA: Princeton Univ. Press, 1970.

[16] Y. Nesterov, "Introductory lectures on convex optimization," in *Applied Optimization*. Boston, MA, USA: Kluwer, 2004.

[17] S. Shahrampour and A. Jadbabaie, "Distributed online optimization in dynamic environments using mirror descent," *IEEE Trans. Autom. Control*, vol. 63, no. 3, pp. 714–725, Mar. 2018.

[18] A. Bernstein, E. Dall'Anese, and A. Simonetto, "Online primal-dual methods with measurement feedback for time-varying convex optimization," *IEEE Trans. Signal Process.*, vol. 67, no. 8, pp. 1978–1991, Apr. 2019.

[19] A. L. Dontchev and R. T. Rockafellar, *Implicit Functions Solution Mappings*. New York, NY, USA: Springer, 2009.

[20] N. Bastianello, A. Simonetto, and R. Carli, "Prediction-correction splittings for time-varying optimization with intermittent observations," *IEEE Control Syst. Lett.*, vol. 4, no. 2, pp. 373–378, Apr. 2020.

[21] O. Besbes, Y. Gur, and A. Zeevi, "Non-stationary stochastic optimization," *Operations Res.*, vol. 63, no. 5, pp. 1227–1244, Oct. 2015.

[22] Y. Li, G. Qu, and N. Li, "Using predictions in online optimization with switching costs: A fast algorithm and a fundamental limit," in *Proc. Annu. Amer. Control Conf. (ACC)*, Jun. 2018, pp. 3008–3013.

[23] A. Rakhlin and K. Sridharan, "Online learning with predictable sequences," in *Proc. 26th Annu. Conf. Learn. Theory, PMLR*, vol. 30, 2013, pp. 993–1019.

[24] A. Jadbabaie, A. Rakhlin, S. Shahrampour, and K. Sridharan, "Online optimization: Competing with dynamic comparators," in *Proc. 18th Int. Conf. Artif. Intell. Statist. (PMLR)*, no. 38, pp. 398–406, 2015.

[25] O. Dekel, N. Haghtalab, and P. Jaillet, "Online learning with a hint," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017.

[26] C.-K. Chiang *et al.*, "Online optimization with gradual variations," in *Proc. 25th Annu. Conf. Learn. Theory, PMLR*, vol. 23, 2012, pp. 6.1–6.20.

[27] A. Simonetto, A. Mokhtari, A. Koppel, G. Leus, and A. Ribeiro, "A class of prediction-correction methods for time-varying convex optimization," *IEEE Trans. Signal Process.*, vol. 64, no. 17, pp. 4576–4591, Sep. 2016.

[28] A. Simonetto, "Dual prediction–correction methods for linearly constrained time-varying convex programs," *IEEE Trans. Autom. Control*, vol. 64, no. 8, pp. 3355–3361, Aug. 2019.

[29] Z. Qi and Y. Zhang, "New models for future problems solving by using ZND method, correction strategy and extrapolation formulas," *IEEE Access*, vol. 7, pp. 84536–84544, 2019.

[30] J. Eckstein, "Splitting methods for monotone operators with applications to parallel optimization," Ph.D. dissertation, Dept. Civil Eng., MIT, Cambridge, MA, USA, Jun. 1989.

[31] L. Jin and Y. Zhang, "Continuous and discrete zhang dynamics for real-time varying nonlinear optimization," *Numer. Algorithms*, vol. 73, no. 1, pp. 115–140, Sep. 2016.

[32] B. Liao, Y. Zhang, and L. Jin, "Taylor $O(h^3)$

discretization of ZNN models for dynamic equality-constrained quadratic programming with application to manipulators," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 2, pp. 225–237, Feb. 2016.

[33] D. Guo, X. Lin, Z. Su, S. Sun, and Z. Huang, "Design and analysis of two discrete-time ZD algorithms for time-varying nonlinear minimization," *Numer. Algorithms*, vol. 77, no. 1, pp. 23–36, Jan. 2018.

[34] B. Qiu, Y. Zhang, J. Guo, Z. Yang, and X. Li, "New five-step DTZD algorithm for future nonlinear minimization with quartic steady-state error pattern," *Numer. Algorithms*, vol. 81, no. 3, pp. 1043–1065, Jul. 2019.

[35] A. S. Charles, A. Balavoine, and C. J. Rozell, "Dynamic filtering of time-varying sparse signals via ℓ_1 minimization," *IEEE Trans. Signal Process.*, vol. 64, no. 21, pp. 5644–5656, Jun. 2016.

[36] A. Y. Popkov, "Gradient methods for nonstationary unconstrained optimization problems," *Autom. Remote Control*, vol. 66, no. 6, pp. 883–891, Jun. 2005.

[37] V. Kungurtsev and J. Jäschke, "A predictor-corrector path-following algorithm for dual-degenerate parametric optimization problems," *SIAM J. Optim.*, vol. 27, no. 1, pp. 538–564, Jan. 2017.

[38] E. L. Allgower and K. Georg, *Numerical Continuation Methods: An Introduction*. New York, NY, USA: Springer-Verlag, 1990.

[39] P. Gong, F. Chen, and W. Lan, "Time-varying convex optimization for double-integrator dynamics over a directed network," in *Proc. Chin. Control Conf.*, 2016, pp. 7341–7346.

[40] C. Xi and U. A. Khan, "Distributed dynamic optimization over directed graphs," in *Proc. IEEE Conf. Decis. Control (CDC)*, Las Vegas, NV, USA, 2016, pp. 245–250.

[41] C. Sun, M. Ye, and G. Hu, "Distributed time-varying quadratic optimization for multiple agents under undirected graphs," *IEEE Trans. Autom. Control*, vol. 62, no. 7, pp. 3687–3694, Jul. 2017.

[42] H. K. Khalil, *Nonlinear Systems*. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.

[43] Y. Zhao and M. N. S. Swamy, "A novel technique for tracking time-varying minimum and its applications," in *Proc. Conf. IEEE Can. Conf. Electr. Comput. Eng.*, vol. 2, May 1998, pp. 910–913.

[44] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *J. Oper. Res. Soc.*, vol. 49, no. 3, pp. 237–252, Apr. 1998.

[45] S. H. Low, F. Paganini, and J. C. Doyle, "Internet congestion control," *IEEE Control Syst.*, vol. 22, no. 1, pp. 28–43, Feb. 2002.

[46] A. Mehrabi, M. Siekkinen, and A. Ylä-Jääski, "QoE-traffic optimization through collaborative edge caching in adaptive mobile video streaming," *IEEE Access*, vol. 6, pp. 52261–52276, 2018.

[47] W. Su, "Traffic engineering and time-varying convex optimization," Ph.D. dissertation, Dept. Elect. Eng., Pennsylvania State Univ., Pennsylvania, PA, USA, May 2009.

[48] M. Maros, "Distributed optimization in time-varying environments," Ph.D. dissertation, School Elect. Eng., KTH, Stockholm, Sweden, 2019.

[49] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, Jun. 2000.

[50] J. L. Jerez, P. J. Goulart, S. Richter, G. A. Constantinides, E. C. Kerrigan, and M. Morari, "Embedded online optimization for model predictive control at megahertz rates," *IEEE Trans. Autom. Control*, vol. 59, no. 12, pp. 3238–3251, Dec. 2014.

[51] J.-H. Hours and C. N. Jones, "A parametric nonconvex decomposition algorithm for real-time and distributed NMPC," *IEEE Trans. Autom. Control*, vol. 61, no. 2, pp. 287–302, Feb. 2016.

[52] B. Gutjahr, L. Gröll, and M. Werling, "Lateral vehicle trajectory optimization using constrained linear time-varying MPC," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 6, pp. 1586–1595, Jun. 2016.

[53] D. Liao-McPherson, M. M. Nicotra, and I. V. Kolmanovsky, "A semismooth predictor corrector method for real-time constrained parametric optimization with applications in model predictive control," in *Proc. IEEE Conf. Decis. Control (CDC)*, Miami Beach, FL, USA, 2018, pp. 3600–3607.

[54] Y. Zhao and W. Lu, "Training neural networks with time-varying optimization," in *Proc. Int. Conf. Neural Netw.*, 1993, pp. 1693–1696.

[55] H. Myeong and J.-H. Kim, "Neural network learning using time-varying two-phase optimization," in *Proc. 33rd IEEE Conf. Decis. Control*, 1994, pp. 1881–1882.

[56] M. Colombino, E. Dall'Anese, and A. Bernstein, "Online optimization as a feedback controller: Stability and tracking," *IEEE Trans. Control Netw. Syst.*, vol. 7, no. 1, pp. 422–432, Mar. 2020.

[57] T. Zheng, J. Simpson-Porco, and E. Mallada, "Implicit trajectory planning for feedback linearizable systems: A time-varying optimization approach," 2019, *arXiv:1910.00678*. [Online]. Available: http://arxiv.org/abs/1910.00678

[58] K.-D. Kim and P. R. Kumar, "Cyber–physical systems: A perspective at the centennial," *Proc. IEEE*, vol. 100, no. Special Centennial Issue, pp. 1287–1308, May 2012.

[59] J. A. Taylor, S. V. Dhople, and D. S. Callaway, "Power systems without fuel," *Renew. Sustain. Energy Rev.*, vol. 57, pp. 1322–1336, May 2016.

[60] A. Bernstein, L. Reyes-Chamorro, J.-Y. Le Boudec, and M. Paolone, "A composable method for real-time control of active distribution networks with explicit power setpoints. Part I: Framework," *Electr. Power Syst. Res.*, vol. 125, pp. 254–264, Aug. 2015.

[61] E. Dall'Anese and A. Simonetto, "Optimal power flow pursuit," *IEEE Trans. Smart Grids*, vol. 9, no. 2, pp. 942–952, Mar. 2018.

[62] A. Hauswirth, A. Zanardi, S. Bolognani, F. Dörfler, and G. Hug, "Online optimization in closed loop on the power flow manifold," in *Proc. IEEE PowerTech Conf.*, Jun. 2017, pp. 1–6.

[63] Y. Tang, K. Dvijotham, and S. Low, "Real-time optimal power flow," *IEEE Trans. Smart Grid*, vol. 8, no. 6, pp. 2963–2973, Nov. 2017.

[64] H. J. Liu, W. Shi, and H. Zhu, "Decentralized dynamic optimization for power network voltage control," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 3, no. 3, pp. 568–579, Sep. 2017.

[65] J. Liu, J. Marecek, A. Simonetto, and M. Takac, "A coordinate-descent algorithm for tracking solutions in time-varying optimal power flows," in *Proc. Power Syst. Comput. Conf. (PSCC)*, Dublin, 2018, pp. 1–7.

[66] Y. Tang, E. Dall'Anese, A. Bernstein, and S. H. Low, "A feedback-based regularized primal-dual gradient method for time-varying nonconvex optimization," in *Proc. IEEE Conf. Decis. Control (CDC)*, Dec. 2018, pp. 3244–3250.

[67] J. Song, E. Dall'Anese, A. Simonetto, and H. Zhu, "Dynamic distribution state estimation using synchrophasor data," *IEEE Trans. Smart Grid*, vol. 11, no. 1, pp. 821–831, Jan. 2020.

[68] M. Gendreau, G. Ghiani, and E. Guerriero, "Time-dependent routing problems: A review," *Comput. Oper. Res.*, vol. 64, pp. 189–197, Dec. 2015.

[69] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus, "On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 3, pp. 462–467, Jan. 2017.

[70] A. Simonetto, J. Monteil, and C. Gambella, "Real-time city-scale ridesharing via linear assignment problems," *Transp. Res. C, Emerg. Technol.*, vol. 101, pp. 208–232, Apr. 2019.

[71] J. Alonso-Mora, A. Wallar, and D. Rus, "Predictive routing for autonomous mobility-on-demand systems with ride-sharing," in *Proc. Conf. Robot. Intell. Syst.*, 2017, pp. 3583–3590.

[72] E. Eser, J. Monteil, and A. Simonetto, "On the tracking of dynamical optimal meeting points," in *Proc. 15th IFAC Symp. Control Transp. Syst.*, Savona, Italy, Jun. 2018, pp. 434–439.

[73] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, Jan. 2009.

[74] A. Balavoine, J. Romberg, and C. Rozell, "Discrete and continuous iterative soft thresholding with a dynamic input," *IEEE Trans. Signal Process.*, vol. 63, no. 12, pp. 3165–3176, Jun. 2015.

[75] Y. Yang, M. Zhang, M. Pesavento, and D. P. Palomar, "An online parallel and distributed algorithm for recursive estimation of sparse signals," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 2, no. 3, pp. 290–305, May 2016.

[76] N. Vaswani and J. Zhan, "Recursive recovery of sparse signal sequences from compressive measurements: A review," *IEEE Trans. Signal Process.*, vol. 64, no. 13, pp. 3523–3549, Jul. 2016.

[77] M. S. Asif and J. Romberg, "Sparse recovery of streaming signals using ℓ_1-homotopy," *IEEE Trans. Signal Process.*, vol. 62, no. 16, pp. 4209–4223, Aug. 2014.

[78] S. M. Fosson, "Online optimization in dynamic environments: A regret analysis for sparse problems," in *Proc. Conf. Decis. Control (CDC)*, Dec. 2018, pp. 7225–7230.

[79] S. Das, P. Lade, and S. Srinivasan, "Model adaptation and unsupervised learning with non-stationary batch data under smooth concept drift," in *Proc. NIPS Time Ser. Workshop*, 2016, pp. 1–11.

[80] M. Maros and J. Jalden, "ADMM for distributed dynamic beamforming," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 4, no. 2, pp. 220–235, Jun. 2018.

[81] Q. Ling and A. Ribeiro, "Decentralized dynamic optimization through the alternating direction method of multipliers," *IEEE Trans. Signal Process.*, vol. 62, no. 5, pp. 1185–1197, Dec. 2014.

[82] F. Y. Jakubiec and A. Ribeiro, "D-MAP: Distributed maximum a posteriori probability estimation of dynamic systems," *IEEE Trans. Signal Process.*, vol. 61, no. 2, pp. 450–466, Jan. 2013.

[83] H. C. M. Clogenson and J. J. van den Dobbelsteen, "Catheters and guide wires for interventional MRI: Are we there yet?" *J. Imag. Interventional Radiol.*, vol. 2, no. 1, p. 28, 2016.

[84] D. Rueckert and J. A. Schnabel, "Model-based and data-driven strategies in medical image computing," *Proc. IEEE*, vol. 108, no. 1, pp. 110–124, Jan. 2020.

[85] S. Zhang, M. Uecker, D. Voit, K.-D. Merboldt, and J. Frahm, "Real-time cardiovascular magnetic resonance at high temporal resolution: Radial FLASH with nonlinear inverse reconstruction," *J. Cardiovascular Magn. Reson.*, vol. 12, no. 1, p. 39, 2010.

[86] S. G. Lingala, B. P. Sutton, M. E. Miquel, and K. S. Nayak, "Recommendations for real-time speech MRI," *J. Magn. Reson. Imag.*, vol. 43, no. 1, pp. 28–44, Jan. 2016.

[87] Q. T. Dinh, C. Savorgnan, and M. Diehl, "Adjoint-based predictor-corrector sequential convex programming for parametric nonlinear optimization," *SIAM J. Optim.*, vol. 22, no. 4, pp. 1258–1284, Jan. 2012.

[88] Y. Nesterov, "Towards non-symmetric conic optimization," *Optim. Methods Softw.*, vol. 27, nos. 4–5, pp. 893–917, Oct. 2012.

[89] D. Kinderlehrer and G. Stampacchia, *An Introduction to Variational Inequalities and Their Applications*. New York, NY, USA: Academic, 1980.

[90] P. Daniele, "Time-Dependent spatial price

equilibrium problem: Existence and stability results for the quantity formulation model," *J. Global Optim.*, vol. 28, nos. 3–4, pp. 283–295, Apr. 2004.

[91] M. G. Cojocaru, P. Daniele, and A. Nagurney, "Projected dynamical systems and evolutionary variational inequalities via Hilbert spaces with applications1," *J. Optim. Theory Appl.*, vol. 127, no. 3, pp. 549–563, Dec. 2005.

[92] A. Nagurney and J. Pan, "Evolution variational inequalities and projected dynamical systems with application to human migration," *Math. Comput. Model.*, vol. 43, nos. 5–6, pp. 646–657, Mar. 2006.

[93] B. Chachuat, B. Srinivasan, and D. Bonvin, "Adaptation strategies for real-time optimization," *Comput. Chem. Eng.*, vol. 33, no. 10, pp. 1557–1567, Oct. 2009.

[94] J. Jäschke and S. Skogestad, "NCO tracking and self-optimizing control in the context of real-time optimization," *J. Process Control*, vol. 21, no. 10, pp. 1407–1416, Dec. 2011.

[95] J. E. A. Graciano, J. Jäschke, G. A. C. Le Roux, and L. T. Biegler, "Integrating self-optimizing control and real-time optimization using zone control MPC," *J. Process Control*, vol. 34, pp. 35–48, Oct. 2015.

[96] O. Arslan and D. E. Koditschek, "Exact robot navigation using power diagrams," in *Proc. ICRA*, 2016, pp. 1–8.

[97] O. Arslan and D. E. Koditschek, "Sensor-based reactive navigation in unknown convex sphere worlds," *Int. J. Robot. Res.*, vol. 38, nos. 2–3, pp. 196–223, Mar. 2019.

[98] M. M. Zavlanos, A. Ribeiro, and G. J. Pappas, "Network integrity in mobile robotic networks," *IEEE Trans. Autom. Control*, vol. 58, no. 1, pp. 3–18, Jan. 2013.

[99] P. Miao, Y. Shen, Y. Huang, and Y.-W. Wang, "Solving time-varying quadratic programs based on finite-time Zhang neural networks and their application to robot tracking," *Neural Comput. Appl.*, vol. 26, no. 3, pp. 693–703, Apr. 2015.

[100] J. Li, M. Mao, F. Uhlig, and Y. Zhang, "Z-type neural-dynamics for time-varying nonlinear optimization under a linear equality constraint with robot application," *J. Comput. Appl. Math.*, vol. 327, no. 1, pp. 155–166, Jan. 2018.

[101] Y. Zhang and C. Yi, *Zhang Neural Networks and Neural-Dynamic Method*. Hauppauge, NY, USA: Nova Science, 2011.

[102] Y. Zhang, L. Jin, D. Guo, Y. Yin, and Y. Chou, "Taylor-type 1-step-ahead numerical differentiation rule for first-order derivative approximation and ZNN discretization," *J. Comput. Appl. Math.*, vol. 273, pp. 29–40, Jan. 2015.

[103] Y. Zhang, Z. Qi, B. Qiu, M. Yang, and M. Xiao, "Zeroing neural dynamics and models for various time-varying problems solving with ZLSF models as minimization-type and euler-type special cases [Research Frontier]," *IEEE Comput. Intell. Mag.*, vol. 14, no. 3, pp. 52–60, Aug. 2019.

[104] A. Simonetto, A. Koppel, A. Mokhtari, G. Leus, and A. Ribeiro, "Decentralized prediction-correction methods for networked time-varying convex optimization," *IEEE Trans. Autom. Control*, vol. 62, no. 11, pp. 5724–5738, Nov. 2017.

[105] Y. Tang, E. Dall'Anese, A. Bernstein, and S. Low, "Running primal-dual gradient method for time-varying nonconvex problems," 2018, *arXiv:1812.00613*. [Online]. Available: http://arxiv.org/abs/1812.00613

[106] S. Fattahi, C. Josz, R. Mohammadi, J. Lavaei, and S. Sojoudi, "Absence of spurious local trajectories in time-varying optimization," 2019, *arXiv:1905.09937*. [Online]. Available: http://arxiv.org/abs/1905.09937

[107] A. Akhriev, J. Marecek, and A. Simonetto, "Pursuit of low-rank models of time-varying matrices robust to sparse and measurement noise," in *Proc. AAAI*, 2020, pp. 1–8.

[108] A. K. Flaxman, A. T. Kalai, and H. McMahan, "Online convex optimization in the bandit setting: Gradient descent without gradient," in *Proc. ACM-SIAM Symp. Discrete Algorithms*, Vancouver, BC, Canada, Jan. 2005, pp. 385–394.

[109] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, "Information-theoretic regret bounds for Gaussian process optimization in the bandit setting," *IEEE Trans. Inf. Theory*, vol. 58, no. 5, pp. 3250–3265, May 2012.

[110] A. Slivkins and E. Upfal, "Adapting to a changing environment: The Brownian restless bandits," in *Proc. COLT*, 2008, pp. 1–12.

[111] I. Shames, D. Selvaratnam, and J. H. Manton, "Online optimization using zeroth order oracles," *IEEE Control Syst. Lett.*, vol. 4, no. 1, pp. 31–36, Jan. 2020.

[112] I. Bogunovic, J. Scarlett, and V. Cevher, "Time-varying Gaussian process bandit optimization," in *Proc. AISTATS*, 2016, pp. 314–323.

[113] N. Bastianello, A. Ajalloeian, and E. Dall'Anese, "Distributed and inexact proximal gradient method for online convex optimization," 2020, *arXiv:2001.00870*. [Online]. Available: http://arxiv.org/abs/2001.00870

[114] E. Dall'Anese, A. Simonetto, S. Becker, and L. Madden, "Optimization and learning with information streams: Time-varying algorithms and applications," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 71–83, May 2020.

[115] K. Yuan, W. Xu, and Q. Ling, "Can primal methods outperform primal-dual methods in decentralized dynamic optimization?" 2020, *arXiv:2003.00816*. [Online]. Available: http://arxiv.org/abs/2003.00816

[116] M. Nonhoff and M. A. Müller, "Online gradient descent for linear dynamical systems," 2019, *arXiv:1912.09311*. [Online]. Available: http://arxiv.org/abs/1912.09311

[117] N. Agarwal, E. Hazan, and K. Singh, "Logarithmic regret for online control," in *Proc. NeurIPS*, 2019, pp. 10175–10184.

[118] S. Agrawal, S. Bubeck, and A. Malek. (Oct. 2020). *Berkeley Simons Institute Program: Theory of Reinforcement Learning, Workshop: Mathematics of Online Decision Making*. Accessed: Mar. 2020. [Online]. Available: https://simons.berkeley.edu/workshops/rl-2020-2

[119] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2017.

[120] T. Chen, S. Barbarossa, X. Wang, G. B. Giannakis, and Z.-L. Zhang, "Learning and management for Internet of Things: Accounting for adaptivity and scalability," *Proc. IEEE*, vol. 107, no. 4, pp. 778–796, Apr. 2019.

[121] B. Li, T. Chen, and G. B. Giannakis, "Secure mobile edge computing in IoT via collaborative online learning," *IEEE Trans. Signal Process.*, vol. 67, no. 23, pp. 5922–5935, Dec. 2019.

ABOUT THE AUTHORS

Andrea Simonetto (Member, IEEE) received the Ph.D. degree in systems and control from Delft University of Technology, Delft, The Netherlands, in 2012.

He spent 3 + 1 years as a Postdoctoral Researcher, first in the Signal Processing Group, Electrical Engineering Department, Delft University of Technology, then in the Applied Mathematics Department, Université catholique de Louvain, Ottignies-Louvain-la-Neuve, Belgium. He joined IBM Research Ireland, Dublin, Ireland, in February 2017, where he is currently a Research Staff Member with the Optimization and Control Group. His research interests include optimization, control, and signal processing, with applications in smart energy, transportation, and personalized health.

Emiliano Dall'Anese (Member, IEEE) received the Ph.D. degree in information engineering from the Department of Information Engineering, University of Padua, Padua, Italy, in 2011.

From January 2009 to September 2010, he was a Visiting Scholar with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN, USA. From January 2011 to November 2014, he was a Postdoctoral Associate with the Department of Electrical and Computer Engineering, University of Minnesota, and from December 2014 to July 2018, he was a Senior Researcher with the National Renewable Energy Laboratory, Golden, CO, USA. He is currently an Assistant Professor with the Department of Electrical, Computer, and Energy Engineering, University of Colorado Boulder, Boulder, CO, USA. His research interests include optimization, control, and signal processing, with applications to networked systems and energy systems.

Santiago Paternain (Member, IEEE) received the B.Sc. degree in electrical engineering from the Universidad de la República Oriental del Uruguay, Montevideo, Uruguay, in 2012, the M.Sc. degree in statistics from The Wharton School of the University of Pennsylvania, Philadelphia, PA, USA, in 2018, and the Ph.D. degree in electrical and systems engineering from the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, in 2018.

His research interest includes optimization and control of dynamical systems.

Dr. Paternain was a recipient of the 2017 CDC Best Student Paper Award.

Geert Leus (Fellow, IEEE) received the M.Sc. and Ph.D. degrees in electrical engineering from Katholieke Universiteit Leuven, Leuven, Belgium, in June 1996 and May 2000, respectively.

He is currently an "Antoni van Leeuwenhoek" Full Professor with the Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Delft, The Netherlands. His research interest includes signal processing, with a specific focus on wireless communications, array processing, sensor networks, and graph signal processing.

Dr. Leus is a Fellow of EURASIP. He was a Member-at-Large of the Board of Governors of the IEEE Signal Processing Society and a member of the IEEE Sensor Array and Multichannel Technical Committee. He is also a member of the IEEE Signal Processing Theory and Methods Technical Committee and the IEEE Big Data Special Interest Group. He received the 2002 IEEE Signal Processing Society Young Author Best Paper Award and the 2005 IEEE Signal Processing Society Best Paper Award. He was the Chair of the IEEE Signal Processing for Communications and Networking Technical Committee. He is the Chair of the EURASIP Special Area Team on Signal Processing for Multisensor Systems. He was the Editor-in-Chief of the *EURASIP Journal on Advances in Signal Processing*. He was also on the Editorial Boards of the IEEE TRANSACTIONS ON SIGNAL PROCESSING, the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, the IEEE SIGNAL PROCESSING LETTERS, and the *EURASIP Journal on Advances in Signal Processing*. He is an Associate Editor of *Foundations and Trends in Signal Processing* and the Editor-in-Chief of *EURASIP Journal on Advances in Signal Processing*.

Georgios B. Giannakis (Fellow, IEEE) received the Diploma degree in electrical engineering. from the National Technical University of Athens, Zografou, Greece, in 1981, and the M.Sc. degree in electrical engineering, the M.Sc. degree in mathematics, and the Ph.D. degree in electrical engineering from the University of Southern California (USC), Los Angeles, CA, USA, in 1983, 1986, and 1986, respectively.

He was a Faculty Member with the University of Virginia, Charlottesville, VA, USA, from 1987 to 1998, and since 1999, he has been a Professor with the University of Minnesota, Minneapolis, MN, USA, where he holds an Endowed Chair, a University of Minnesota McKnight Presidential Chair in Electrical and Computer Engineering, and serves as the Director of the Digital Technology Center. He has published more than 465 journal articles, 765 conference papers, 25 book chapters, two edited books, and two research monographs. He is the (co-) inventor of 33 issued patents. His general research interests include statistical learning, communications, and networking. His current research interests include data science and network science with applications to the Internet of Things and power networks with renewables.

Dr. Giannakis is a Fellow of the National Academy of Inventors, the European Academy of Sciences, and EURASIP. He was a (co-) recipient of nine best journal paper awards from the IEEE Signal Processing (SP) and Communications Societies, including the G. Marconi Prize Paper Award in wireless communications. He also received the IEEE-SPS Nobert Wiener Society Award in 2019, the EURASIP's A. Papoulis Society Award in 2020, the Technical Achievement Awards from the IEEE-SPS in 2000 and from EURASIP in 2005, the IEEE ComSoc Education Award in 2019, the G. W. Taylor Award for Distinguished Research from the University of Minnesota, and the IEEE Fourier Technical Field Award in 2015. He has served the IEEE in a number of posts, including that of a Distinguished Lecturer for the IEEE-SPS.