Stochastic Gradient Methods for Stochastic Model Predictive Control

Andreas Themelis, Silvia Villa, Panagiotis Patrinos, Alberto Bemporad

***Abstract*— We introduce a new stochastic gradient algorithm, SAAGA, and investigate its employment for solving Stochastic MPC problems and multi-stage stochastic optimization programs in general. The method is particularly attractive for scenario-based formulations that involve a large number of scenarios, for which "batch" formulations may become inefficient due to high computational costs. Benefits of the method include cheap computations per iteration and fast convergence due to the sparsity of the proposed problem decomposition.**

I. Introduction

In the field of control theory, the state of the art methodology for dynamic decisions is Model Predictive Control (MPC), which faces infinite-horizon constrained optimal control problems by computing solutions of an approximate finite-horizon problem at each sampling time, and then implementing the control law in accordance to a receding horizon strategy. Initially restricted to the control of petroleum refineries in the end of the '70s, MPC is now employed in a wide variety of areas ranging from engineering to financial applications (see e.g. [1] and the references therein).

Stochastic Model Predictive Control (SMPC) includes the dependency on *random* events. The uncertainty can either come from data intrinsic to the problem such as random demand, resources at disposal, noises, or possibly from modeling errors.

Randomness is often assumed to have a convenient known distribution, typically Gaussian [2], [3], [4], but the assumption may be too restrictive. An alternative is to approximate the stochastic process by a process having finitely many sample paths (called scenarios) and exhibiting a tree structure that models its filtration [5], [6], [7]. Because of the exponential growth of scenario trees with respect to time horizon and probability distribution, solving SMPC problems soon becomes very challenging in terms of computational requirements. One solution is developing algorithms amenable to parallelization (see [8] and the references therein).

With the purpose of minimizing computational costs, in the present paper we discuss stochastic first order approaches for SMPC problems on scenario trees. More precisely, we derive a new stochastic method for solving SMPC by combining two recent ideas used to improve convergence behavior of stochastic first order methods: aggregation of stochastic gradients [9] and variable metric approaches [10]. The idea of using stochastic algorithms in this context was already mentioned in [11] but to the best of our knowledge no significant advances have been made so far. Proximal Stochastic Gradient methods (SG) are often used to minimize functions of the form

$$F(x) \;=\; f(x)+\varphi(x) \coloneqq \frac{1}{n}\sum_{i=1}^{n} f_i(x)+\varphi(x) \qquad (1)$$

where $\varphi : \mathbb{R}^N \to \mathbb{R}\cup\{+\infty\}$ is convex and each of the component functions $f_i\colon \mathbb{R}^N \to \mathbb{R}$ is convex and smooth on $\mathbf{dom}\,\varphi$. Differently from standard Gradient algorithms, which require the computation of the full gradient of the smooth part at each iteration, SG requires the gradient of only one component f_i. The complexity per iteration is then independent of n, which enables the possibility to handle large problems, without waiting to process all the components before updating the solution.

The idea is then to decompose the expected cost of a SMPC problem or, more generally, of a stochastic multistage program, as

$$f(x) \;=\; \sum_{i=1}^{n} p_i f_i(x) \qquad (2)$$

where f_i is the cost associated to the i-th *scenario* and p_i is the probability it occurs, and to assign possible constraints to the nonsmooth term φ. Sampling indices $i = 1 \dots n$ according to the intrinsic distribution of the problem $p_1 \dots p_n$ yields a random *unbiased* selection of the gradients: $\mathbb{E}[\nabla f_i(x)] = \nabla f(x)$. If we are able to simulate (or *observe*) i.i.d. scenario realizations, then we do not need to know what the actual values of p_i are. This conforms to real world applications where the stochastic framework is likely unknown. Moreover, since stochastic methods optimize the system using the information of the sampled function, from an optimization point of view this sampling criterion benefits from *biasing* the optimization so to favour the most likely events. Most importantly, we can provide *explicit* stopping criteria tailored for closed loop formulations.

The paper is organized as follows: in §II we provide an overview on SG methods and introduce some notation. In §III we present our stochastic algorithm, SAAGA, and we outline the main steps for its convergence proof. In §IV we formally define scenario trees and the SMPC problem at hand, and in §V we discuss how to solve it with SAAGA. In §VI with some simulations we show how our method significantly improves the state-of-the-art algorithms it is derived from. Finally, in §VII we draw some conclusions and motivate future research plans.

II. Stochastic Gradient methods

In this section we provide a brief overview on the main ingredients of our analysis: aggregated SG methods and vari-

able metric approaches. The interested reader is referred to [12] for the basic definitions and notation.

A. Aggregated Stochastic Gradient methods

The standard (proximal) Stochastic Gradient Descent method (SGD) for minimizing (1) prescribes updates of the form $x^k = \mathbf{prox}_{\gamma_k\varphi}\big(x^{k-1} - \gamma_k \nabla f_{j_k}(x^{k-1})\big)$ where index j_k is *sampled* in $\{1 \dots n\}$ in either a cyclic, shuffled, or random fashion, and the stepsize γ_k suitably decreases to zero. Known drawbacks include the issue of suitably choosing the stepsize, the slow convergence of the method, and instability when close to a solution due to the non monotonic behaviour.

Many variants have been proposed in order to improve the convergence, such as including momentum terms or averaging iterates or gradients. In this paper, we focus on the idea of *aggregating* the sampled gradient to some of the previously computed ones, hence involving direction of descent of more component functions at once yet mantaining the cheap computational effort of SGD [13], [14], [9], [15], [16]. We will refer to this class of methods as *Aggregated Stochastic Gradient* methods (AgSG). Even if there are differences among the various AgSG algorithms, e.g. in the choice of the components to be sampled, such variants exhibit faster convergence rates with respect to SGD, and, more importantly, boast the employment of a constant step size. Intuitively, the aggregated gradient approximates the full gradient, so that the algorithm behaves well even when close to the solution. In particular, SAGA [9] allows for the presence of a nonsmooth term φ. The direction of descent at step k is given by

$$d^k_{\text{SAGA}} \;=\; \nabla f_{j_k}(x^k) - y_{j_k} + \frac{1}{n}\sum_{i=1}^{n} y_i \tag{3}$$

where j_k is the sampled index and y_i is the gradient computed the last time in the past index i was sampled.

B. Variable metric selection

With *metric* we refer to a symmetric and positive definite matrix $\mathbf{H}$ and to the induced scalar product $\langle x, y\rangle_{\mathbf{H}} := \langle x, \mathbf{H}y\rangle$. With $\|\cdot\|_{\mathbf{H}}$ we denote the induced norm, and with $\mathbf{prox}^{\mathbf{H}}_{\varphi}$ the proximity operator of φ with respect to $\|\cdot\|_{\mathbf{H}}$

$$\mathbf{prox}^{\mathbf{H}}_{\varphi}(x) := \operatorname*{\mathbf{arg\,min}}_z \Big\{\varphi(z) + \tfrac{1}{2}\|z - x\|^2_{\mathbf{H}}\Big\}$$

or equivalently, letting $\widetilde{x} = \mathbf{H}^{1/2}x$ and $\widetilde{\varphi} = \varphi \circ \mathbf{H}^{-1/2}$,

$$= \mathbf{H}^{-1/2}\,\mathbf{prox}_{\widetilde{\varphi}}(\widetilde{x}) \tag{4}$$

When minimizing (1), operating in the metric induced by $\mathbf{H}$ boils down to considering the scaled problem $\widetilde{f}(\widetilde{x}) + \widetilde{\varphi}(\widetilde{x})$. It is well known that first order methods are particularly sensitive to ill conditioning, which makes the selection of a good scaling $\mathbf{H}$ a crucial issue for performance. For stochastic methods, where only small portions of the function are available at each time, the optimal metric is updated at each iteration giving rise to a variable metric $\{\mathbf{H}_k\}$ (the employment of variable metrics are also considered for full (sub)gradient methods: see e.g. [17]).

In [10] AdaGrad adaptive scaling is introduced to enhance the convergence speed of SGD in the context of online learning and regret minimization. The square of the scaling matrix at time k is recursively defined as

$$\mathbf{A}^2_k \;=\; \begin{cases} \delta^2\mathbf{I} & \text{for } k = 0 \\ \mathbf{A}^2_{k-1} + \mathbf{diag}\big(g^k(g^k)^\intercal\big) & \text{for } k \geq 1 \end{cases} \tag{5}$$

where g^k is the sampled (sub)gradient at time k and δ is any positive scalar. The efficiency of the scaling is motivated in the context of standard SGD iterations applied to regret minimization.

In the next section we consider the idea of combining variable metrics and AgSG algorithms and blend together the respective state-of-the-art methods.

III. SAAGA Algorithm

We now present the SAAGA algorithm, the first 'A' being short for 'adaptive', which combines AgSG method SAGA with a variable metric inspired by AdaGrad. Variable metrics in the context of AgSG methods have already been considered in [15], however under different assumptions on the metric and the sampling strategy. In the following, we provide a sketch of the convergence proof, leaving the details of the analysis to an extended version of this work.

Integrating a variable metric $\{\mathbf{H}_k\}$ in SAGA results in updates of the form

$$x^k \;=\; \mathbf{prox}^{\mathbf{H}_k}_{\gamma\varphi}\big(x^{k-1} - \gamma\mathbf{H}_k^{-1}d^k\big) \tag{6}$$

with d^k as in (3). Extending the computations of [10] and relying on an *online-to-batch* conversion [18] we get that for sparse problems a similar heuristic as (5) enhances convergence. The divergent behaviour of the matrices in (5) yields a vanishing stepsize which is necessary for the convergence of the method. However, for SAGA this is highly penalizing which is why we perform the following "normalization"

$$\mathbf{H}_k \;:=\; \|\mathbf{A}_k\|^{-1}_{\infty}\mathbf{A}_k \tag{7}$$

where $\mathbf{A}_k$ are as in (5) with g^k being the aggregated gradient.

A. Convergence proof

In the following we assume f_i to be L-smooth and μ-strongly convex on $\mathbf{dom}\,\varphi$, and with $\mathbb{E}^k$ we denote the expectation conditional to the knowledge at step k. We define a scaled version of the Lyapunov function considered in [9]:

$$T^k = T(x^k, \{\phi^k_i\}_i) := \frac{1}{n}\sum_{i=1}^{n}\mathcal{D}_{f_i}(\phi^k_i) + c\|x^k - x^\star\|^2_{\mathbf{H}_k} \tag{8}$$

where $c > 0$ is to be defined, ϕ^k_i denotes the iterate x when index i was last sampled before step k, $x^\star$ is the (unique) solution, and $\mathcal{D}_{f_i}(x) := f_i(x) - f_i(x^\star) - \langle\nabla f_i(x^\star), x - x^\star\rangle$. Extending the proof in terms of the primal-dual metric pair $\|\cdot\|_{\mathbf{H}_k}$ / $\|\cdot\|_{\mathbf{H}_k^{-1}}$ and involving the metric residuals $\nu_k =$

Algorithm I SAAGA: SAGA [9] with normalized **A**daptive scaling for minimizing (1).

INITIALIZE: $x^0 \in \mathbf{dom}\,\varphi$; $y_1 \dots y_n \in \mathbb{R}^N$; $0 < h \in \mathbb{R}^n$
$\gamma > 0$; $a \leftarrow 0$.
For iterations $k = 0, 1 \dots$ loop as follows:
1: Sample a random index j
2: Compute $g = \nabla f_j(x^k)$
3: $d \leftarrow g - y_j + \frac{1}{n}\sum_{i=1}^n y_i$ and $y_j \leftarrow g$
4: $a \leftarrow a + (d)^2$ and $h \leftarrow \left({}^{a}/_{\|a\|_\infty}\right)^{1/2}$ % (pointwise)
5: Update $x^{k+1} = \mathbf{prox}_{\gamma\varphi}^{\mathbf{diag}(h)}(x^k - \gamma d/h)$ % (p.wise ratio)

$\|x^k - x^\star\|_{\mathbf{H}_k} - \|x^k - x^\star\|_{\mathbf{H}_{k-1}}$ we end up with the bound

$$\begin{aligned}\mathbb{E}^k\big[T^{k+1}\big] - \big(1 - \tfrac{1}{\kappa}\big)T^k \\ \leq \overbrace{\Big(\tfrac{1}{\kappa} + \tfrac{2cL\gamma^2(1+\beta^{-1})}{\lambda} - \tfrac{1}{n}\Big)}^{C_1}\frac{1}{n}\sum_{i=1}^n \mathcal{D}_{f_i}(\phi_i^k) \\ + \overbrace{\Big(\tfrac{1}{n} - 2c\gamma\Big(\tfrac{\mu\beta\gamma}{\lambda} + \tfrac{L-\mu}{L}\Big)\Big)}^{C_2}\mathcal{D}_f(x^k) \\ + c\gamma\underbrace{\Big(\tfrac{1+\beta}{\lambda}\gamma - \tfrac{1}{L}\Big)}_{C_3}\mathbb{E}^k\big[\|\nabla f_{j_{k+1}}(x^k) - \nabla f_{j_{k+1}}(x^\star)\|^2\big] \\ + c\underbrace{\big(\tfrac{1}{\kappa} - \gamma\mu\big)}_{C_4}\|x^k - x^\star\|^2 + c\mathbb{E}^k[\nu_{k+1}] \qquad (9)\end{aligned}$$

which holds for any $\beta, \gamma, \kappa > 0$ as long as $\lambda_{\min}(\mathbf{H}_k) \geq \lambda$.

Theorem III.1. *Let F be as in (1). Let $L > 0$ and $\mu > 0$. Assume that $\varphi : \mathbb{R}^N \to \overline{\mathbb{R}}$ is proper convex lower semicontinuous and with bounded domain, and that every f_i is L-smooth and μ-strongly convex on $\mathbf{dom}\,\varphi$. Then, the sequence $\{x^k\}$ generated by Algorithm I with $\gamma = 1/3L$ converges a.s. to $x^\star = \mathbf{arg\,min}\, F$.*

Proof. Letting $r := \mu/L \in (0, 1]$, it can be readily verified that $C_i \leq 0\ \forall i$ in (9) by choosing

$$\gamma = \tfrac{1}{3L}, \quad \lambda = \tfrac{2+r}{3}, \quad \kappa = 2n\tfrac{1+r}{r}, \quad \beta = 1 + r, \quad c = 3L\tfrac{2+r}{4n}$$

If $\mathbf{dom}\,\varphi$ is bounded, by smoothness so is the sequence $\{g^k = \nabla f_{j_k}(x^k)\}$ and it can be easily checked that $\lambda_k := \lambda_{\min}(\mathbf{H}_k) \to 1$. In particular, (still by boundedness) $\nu_i \to 0$ surely and $\lambda_k \geq \lambda$ for k large enough. Letting $\rho = 1 - 1/\kappa$ with respect to the *unconditional* expectation we then have

$$\mathbb{E}\big[T^k\big] \leq \rho T^{k-1} + c\mathbb{E}[\nu_k] \leq \cdots \leq \rho^k T^0 + c\sum_{i=0}^k \rho^{k-i}\,\mathbb{E}[\nu_i]$$

and since $\rho \in (0, 1)$, then $c\lambda\|x^k - x^\star\|^2 \leq T^k \to 0$. □

B. Implementation issues

Vectors y_i store the last sampled gradients and can therefore be warm started offline as $\nabla f_i(x^0)$, otherwise initializing at 0 is advisable. The constant L needs not be known in advance as it can be updated with a linesearch involving solely the sampled function f_j: starting from, say, $L = 1$, we double it until it holds that

$$f_j\big(\underline{u} - \tfrac{1}{L}\nabla f_j(\underline{u})\big) - f_j(\underline{u}) \ \leq\ -\tfrac{1}{2L}\|\nabla f_j(\underline{u})\|^2 \qquad (10)$$

IV. Problem formulation

We now describe the investigated MPC problem. Let us consider a state-input dynamical system

$$\begin{cases} x_t = \mathbf{A}_t x_{t-1} + \mathbf{B}_t u_{t-1} \\ x_0 = \bar{x}_\bullet \\ t \ = 1 \dots T \end{cases} \qquad (11)$$

$\bar{x}_\bullet$ is the initial state and for $t = 1 \dots T$ we assume $\mathbf{A}_t$ and $\mathbf{B}_t$ to depend on the realization of a random variable $\xi_t : \Omega_t \to \Xi_t$, whose outcome is known only after the input u_{t-1} is fed to the system. For simplicity we consider a linear system, but an additional affine (stochastic) term c_t in the dynamics could also be taken into account. The choice of input u_{t-1} can be estimated only using the *past* information, that is, the outcomes of $\xi_1 \dots \xi_{t-1}$. Any input selection policy is hence a function $u_t : \Xi_1 \times \cdots \times \Xi_t \to \mathbb{R}^m$ (u_0 is constant), or, in other words, the policy $u_0 \dots u_{T-1}$ is adapted to the filtration $\big(\sigma(\xi_1, \cdots, \xi_{t-1})\big)_{t=1\dots T}$. This is known as *nonanticipativity constraint*.

The *optimality* of a policy $\underline{u} = (u_0 \dots u_{T-1})$ is measured in terms of a cost function

$$f(\underline{u}) \coloneqq \mathbb{E}\left[\sum_{t=0}^{T-1} \ell_t\big(u_t, x_t\big) + \ell_T(x_T)\right] \qquad (12)$$

where $x_0 \dots x_T$ are given by the dynamics (11). Here, $u_t = u_t(\xi_1, \cdots, \xi_t)$ and the expectation in (12) is taken with respect to $\xi_1 \times \cdots \times \xi_T$.

Assumption 1 (Convexity and smoothness)**.** *The per-stage costs ℓ_t are strongly convex and differentiable with L-Lipschitz continuous gradient for some $L \geq 0$.*

For the sake of simplicity we assume the states not to be constrained, but the hypothesis can clearly be relaxed. However, the assumption still embraces many problems such as, for instance, those in which states are *soft-constrained* (see e.g. [19]). In the investigated problems we then assume the *feasibility* of $\underline{u}$ to be enforced by some convex constraints

$$u_t \ \in\ \mathcal{U}_t, \quad t \ =\ 0 \dots T-1 \qquad (13)$$

Assumption 2. *The feasibility sets $\mathcal{U}_t$ $t = 0 \dots T-1$ are nonempty, closed, convex, bounded and easy to project onto.*

Finding an optimal and feasible policy boils down to solving the multistage stochastic optimization problem

$$\underset{(u_t(\xi_1 \dots \xi_t))_{t=0}^{T-1}}{\mathbf{minimize}}\ \mathbb{E}\left[\sum_{t=0}^{T-1} \ell_t\big(u_t, x_t\big) + \ell_T(x_T)\right] \ \text{s.t. } u_t \in \mathcal{U}_t \qquad (14)$$

where x_t satisfy the dynamics (11).

A. Scenario tree

We assume the following:

Assumption 3. *The random variables ξ_t are finitely supported and with known joint distributions.*

Finiteness is a common and quite reasonable requirement, as *scenario reduction* techniques [20] are usually employed in generating finite approximations of a distribution. On the

other hand we will later discuss how to operate in case the random distributions are not known.

By Assumption 3 all the combinations of outcomes of ξ_t can be arranged in a tree structure of height T that we refer to as *scenario tree*. With $\mathcal{N}$ we denote the set of its nodes, and with $\mathcal{N}_t \subseteq \mathcal{N}$ those at level t ($\emptyset$ for $t > T$). For $\nu \in \mathcal{N}_t$ we denote $\tau(\nu)$ as its level t, we define $C(\nu) \subseteq \mathcal{N}_{t+1}$ as the set of its children, and if ν is not the root we refer to its ancestor as $a(\nu)$.

On such tree we define the partial order relation $\geq$ according to which $\nu \geq \mu$ if μ belongs to the path from the root to ν. For $\mu \geq \nu$ we denote with $[\mu, \nu]$ the path from μ up to ν (inclusive of μ and ν); if μ is the root node then we may omit it and write simply $[\nu]$. If instead $\mu > \nu$, then $[\mu, \nu]$ is by convention empty.

Starting from a root node at level 0 we recursively define all the other nodes as follows: for $t = 1 \dots T$ and for any $\nu \in \mathcal{N}_{t-1}$ we assign each of the children nodes $\nu' \in C(\nu)$ a different outcome of ξ_t. Each node $\nu \in \mathcal{N}_t$, $t = 0 \dots T$, is assigned a probability weight p_ν that corresponds to the event that $\xi_1 \dots \xi_t$ have evolved according to the path from the root to the node itself, namely

$$p_\nu \;=\; \mathbb{P}\big[\xi_t \;=\; \nu, \xi_{t-1} \;=\; a(\nu), \xi_{t-2} \;=\; a^2(\nu) \dots\big]$$

The root, which is not assigned any outcome, is denoted with a small bullet '$\bullet$', and is assigned probability $p_\bullet = 1$. If $\nu \in \mathcal{N}_t$ is such that $p_\nu = 0$ then the node itself and the subtree rooted at ν can be discarded. This happens when the conditional probability $\mathbb{P}[\xi_t = \nu \,|\, \xi_{t-1} = a(\nu)]$ is null.

With a slight abuse of notation we may emphasize the correspondence between the nodes and the random outcomes by identifying node $\nu \in \mathcal{N}_t$ with the t-uple $(\xi_1, \cdots, \xi_t)$ (the root then corresponds to the empty tuple).

B. Nodal decomposition

Motivated by the dependencies of the random variables on states and inputs, it is handy to visualize all the possible inputs and states on the scenario tree. To the node $\nu = (\xi_1, \cdots, \xi_t)$ we associate the variables $u_t(\xi_1, \cdots, \xi_t)$ and $x_t(\xi_1, \cdots, \xi_t)$ that we shall refer to simply as u_ν and x_ν, respectively. Clearly, no input is associated to the leaf nodes $\nu \in \mathcal{N}_T$ as well as no state *variable* is defined at the root, in that $x_\bullet$ is rather the constant $\bar{x}_\bullet$.

To streamline the notation we define $\mathcal{N}^x := \mathcal{N} \setminus \mathcal{N}_0$ and $\mathcal{N}^u := \mathcal{N} \setminus \mathcal{N}_T$ respectively as the sets of nodes on which input and node variables are indexed. The system data is indexed consistently as $\mathbf{A}_\nu$ and $\mathbf{B}_\nu$ for $\nu \in \mathcal{N}^x$.

With this notation, we may reformulate (11) as

$$\underset{(u_\nu \in \mathbb{R}^m)_{\nu \in \mathcal{N}^u}}{\mathbf{minimize}} \sum_{t=0}^{T-1} \sum_{\nu \in \mathcal{N}_t} p_\nu \ell_t(u_\nu, x_\nu) + \sum_{\nu \in \mathcal{N}_T} p_\nu \ell_T(x_\nu) \quad (15a)$$

subject to $u_\nu \in \mathcal{U}_{\tau(\nu)}$ and with x_ν recursively defined as

$$\begin{cases} x_\bullet = \bar{x}_\bullet \\ x_\nu = \mathbf{A}_\nu x_{a(\nu)} + \mathbf{B}_\nu u_{a(\nu)} \quad \nu \in \mathcal{N}^x \end{cases} \quad (15b)$$

V. Stochastic methods for MPC

We now switch to the core of the present paper and study how to implement efficient stochastic algorithms tailored for the MPC problem at hand. We start with a general analysis that applies to SG methods in general; in particular, as anticipated we propose a scenario-based decomposition of the cost function and we discuss the advantages of such choice. Then we focus on SAAGA algorithm providing further evidence in support of its employment.

A. Scenario decomposition

For each scenario $[\nu] = (\nu_0 \dots \nu_T)$ or, equivalently, for each leaf node $\nu \in \mathcal{N}_T$, we define

$$f_{[\nu]}(\underline{u}) := \sum_{t=1}^{T} \ell_t(u_{\nu_t}, x_{\nu_t}) + \ell_T(x_{\nu_T}) \quad (16)$$

as the cost function corresponding to the random realization $\{\xi_1 = \nu_1, \cdots, \xi_T = \nu_T\}$. (Here, as usual, $x_{\nu_t} = x_{\nu_t}(\underline{u})$ is recursively defined as in (15b)). Then, the total cost (expectation) can be decomposed as the *weighed* sum

$$f \;=\; \sum_{\nu \in \mathcal{N}_T} p_\nu f_{[\nu]} \quad (17a)$$

The feasibility constraints can be handled in a separable indicator function

$$\varphi(\underline{u}) \;=\; \delta_{\times_{\mu \in \mathcal{N}^u} \mathcal{U}_{\tau(\mu)}}(\underline{u}) \;=\; \sum_{\mu \in \mathcal{N}^u} \delta_{\mathcal{U}_{\tau(\mu)}}(u_\mu) \quad (17b)$$

In the end the problem can be formulated in the shape of (1) as follows:

$$\underset{\underline{u} = (u_\mu)_{\mu \in \mathcal{N}^u}}{\mathbf{minimize}} \; F(\underline{u}) := \overbrace{\sum_{\nu \in \mathcal{N}_T} f_{[\nu]}(\underline{u})}^{f} + \overbrace{\sum_{\mu \in \mathcal{N}^u} \delta_{\mathcal{U}_{\tau(\mu)}}(u_\mu)}^{\varphi} \quad (18)$$

1) Biased sampling for unbiased gradients: We shall underline that p_ν are simply probability weights and are not part of the component functions $f_{[\nu]}$. SAGA and most SG algorithms rely on the unbiasedness of the sampled gradients $\{g^k\}$, that is, the fact that

$$\mathbb{E}^k\big[g^{k+1}\big] \;=\; \nabla f(x^k) \quad (19)$$

Instead of selecting the component function with equal probability we then bias the sampling according to the distribution $\mathbb{P}[j_k = \nu] = p_\nu$ so to ensure (19). As a byproduct, the more probable a scenario the more frequently its cost function is sampled so that the optimization favours likely outcomes and overlooks improbable realizations. Besides, it is not necessary to know the probability weights as the scenarios can be sampled on the fly directly from the real environment.[1]

[1] SAAGA actually requires such knowledge in order to correctly compute the update directions (cf. line 7 of Alg. III), but it is still possible to use estimates and refine them at each sampling à la Monte Carlo: $f_{[\nu]}$ does not depend on p_ν and any refinement would not affect the rest of the algorithm.

Algorithm II Computation of the gradient of the scenario cost function $f_{[\nu]}$.
The expressions under the horizontal brackets refer to the quadratic cost case $\ell_t(x,u) = \frac{1}{2}\|x\|^2_{\mathbf{Q}} + \frac{1}{2}\|u\|^2_{\mathbf{R}}$ $\forall t$.

for $t = 1 \dots T$ **do** % states update
  $x_{\nu_t} \leftarrow \mathbf{A}_{\nu_t} x_{\nu_{t-1}} + \mathbf{B}_{\nu_t} u_{\nu_{t-1}}$
end for
$g^{T-1} \leftarrow \underbrace{\nabla \ell_T(x_{\nu_T})}_{\mathbf{Q}x_{\nu_T}}$
for $t = T-2 \dots 0$ **do**
  $g^t \leftarrow \underbrace{\nabla_x \ell_{t+1}(x_{\nu_{t+1}}, u_{\nu_{t+1}})}_{\mathbf{Q}x_{\nu_{t+1}}} + \mathbf{A}^\intercal_{\nu_{t+2}} g^{t+1}$
end for
for $t = 0 \dots T-1$ **do**
  $g^t \leftarrow \underbrace{\nabla_u \ell_t(x_{\nu_t}, u_{\nu_t})}_{\mathbf{R}u_{\nu_t}} + \mathbf{B}^\intercal_{\nu_{t+1}} g^t$
end for
return $g^t = \nabla_{u_{\nu_t}} f_{[\nu]} \quad t = 0 \dots T-1$

2) Stopping criterion for MPC: Stopping criteria for SG methods are not an easy task since from local information we cannot infer the status of the whole system. However, according to the receding horizon principle the only variable of interest is the first input. Since all the scenarios share the root node, the first input is updated always. It is then easy to monitor the variations of $u^k_\bullet$ so to quit the algorithm when, say, the mean residual on a window of w iterations

$$R^k_w := \frac{1}{w} \sum_{i=0}^{w-1} \|u^{k-i}_\bullet - u^{k-i-1}_\bullet\|^2 \tag{20}$$

falls below a given threshold. Intuitively, this criterion should work particularly well with aggregated gradients for they gradually approximate the full gradient. Indeed, notice that

$$\begin{aligned} \underline{u}^k - \underline{u}^{k-1} &= \mathbf{prox}^{\mathbf{H}_k}_{\gamma\varphi}\big(\underline{u}^{k-1} - \gamma \mathbf{H}_k^{-1} d^k \quad\quad\big) - \underline{u}^{k-1} \\ &\approx \mathbf{prox}^{\mathbf{H}_k}_{\gamma\varphi}\big(\underline{u}^{k-1} - \gamma \mathbf{H}_k^{-1} \nabla f(\underline{u}^{k-1})\big) - \underline{u}^{k-1} \end{aligned}$$

and the rightmost handside being zero is equivalent to the optimality of $\underline{u}^{k-1}$. Being interested in the optimality of the first input only, in (20) we restrict our attention to the corresponding components and discard all the others.

B. SAAGA Algorithm for SMPC

We briefly discuss some issues for implementing the investigated SMPC problem (15) in SAAGA, and then summarize the procedure in Algorithm III. To avoid overcomplicating the exposition we do not include the stopping criterion discussed in §V-A.2.

1) Sparse gradients storage: To keep track of the last computed gradients we need to pre-allocate a space y_ν for each leaf node ν. Each $f_{[\nu]}$ depends only on T many inputs out of the $|\mathcal{N}^u|$ total ones, specifically on those on the scenario path $u_{\nu_0} \dots u_{\nu_{T-1}}$. In particular, $\nabla f_{[\nu]}$ is sparse (with density $O(\log x / x)$ where $x = |\mathcal{N}^u|$). This has two consequences: first, from a convergence speed point of view the contribution of AdaGrad is particularly effective [10], and second, from a memory allocation standpoint we need to store in memory only a logarithmic portion of the entire gradient for each component function $f_{[\nu]}$. Such nonzero components $y^0_\nu \dots y^{T-1}_\nu$ can be computed with the steps given in Algorithm II; y^t_ν will keep track of the gradient $\frac{\partial f_{[\nu]}}{\partial u_{\nu_t}}$.

Algorithm III SAAGA Algorithm for SMPC problem (15)

Initialize: $u_\mu, a_\mu, r_\mu \leftarrow 0_m$ $(\mu \in \mathcal{N}^u)$
$y^t_\nu \leftarrow 0_m$ $(\nu \in \mathcal{N}_T,\ t = 0 \dots T-1)$
$L \leftarrow 1$ % Lipschitz stepsize first estimate
$V \leftarrow \{\text{(root)}\}$ % set of visited nodes
For iterations $k = 0, 1 \dots$ loop as follows:
Sample a random scenario $[\nu] = (\nu_0 \dots \nu_T)$
$V \leftarrow V \cup \{\nu_1, \cdots, \nu_{T-1}\}$
Compute $g_t = \frac{\partial f_{[\nu]}}{\partial u_{\nu_t}}(\underline{u}^k)$, $(t = 0 \dots T-1)$ with Alg. II
Update L with line search (10) on $f_{[\nu]}$
for $t = 0 \dots T-1$ **do** (in parallel)
  $d_{\nu_t} \leftarrow g_t - y^t_\nu + r^t_\nu$ % update direction
  $r_{\nu_t} \leftarrow r_{\nu_t} + p_\nu (g_t - y^t_\nu)$ % update past gradients average
  $y^t_\nu \leftarrow g_t$
end for
for $\mu \in V$ **do** (in parallel)
  $a_{\nu_t} \leftarrow a_{\nu_t} + (g_t)^2$ and $h \leftarrow (a_{\nu_t})^{1/4}$ % metric sq. root
  $u^{k+1}_\mu \leftarrow h^{-1} \odot \mathbf{\Pi}_{h \odot \mathcal{U}_{\tau(\mu)}}\big[h \odot u^k_\mu - h^{-1} \odot d_\mu\big]$
end for

2) Active nodes: Up to when a node μ is not "visited", that is, up to the first time a leaf $\nu > \mu$ is sampled, the corresponding variable u_μ remains unchanged. To speed up early iterations we may keep track of visited nodes in a set V and update variables u_μ only for $\mu \in V$.

3) Mean and drift directions: Consistently with what discussed in §V-A.1, the (unscaled) update direction (3) now is rather given by

$$d^{k+1} = \overbrace{\nabla f_{[\bar\nu]}(x^k) - y_{\bar\nu}}^{\textit{drift}} + \overbrace{\sum_{\nu \in \mathcal{N}^T} p_\nu y_\nu}^{\textit{mean}} \tag{21}$$

where $\bar\nu$ is the sampled scenario. Instead of computing the weighed average from scratch we may store the average in a vector r and update it as $r \leftarrow r + p_{\bar\nu}(\nabla f_{[\bar\nu]}(x^k) - y_{\bar\nu})$ at the end of the iteration.

4) Proximal step: By the definition of φ in (17b) we have that its proximal mapping separates as

$$\mathbf{prox}_{\gamma\varphi}\big((u_\mu)_{\mu \in \mathcal{N}^u}\big) = \bigtimes_{\mu \in \mathcal{N}^u} \mathbf{\Pi}_{\mathcal{U}_{\tau(\mu)}}(u_\mu)$$

which is fully parallelizable on all variables and easy to compute by Assumption 2. Using (4) we obtain $\mathbf{prox}^{\mathbf{H}}_{\delta_{\mathcal{X}}}(x) = h^{-1/2} \odot \mathbf{\Pi}_{h^{1/2} \odot \mathcal{X}}(h^{1/2} \odot x)$ for $\mathbf{H} = \mathbf{diag}(h)$ where $\odot$ denotes the Hadamard (elementwise) product.

VI. Simulations

In this section we provide a preliminary simulation to test the performance of SAAGA for the problem at hand. We consider a small SMPC problem as toy example with the purpose of providing evidence about how the investigated method significantly outperforms both its parent algorithms.

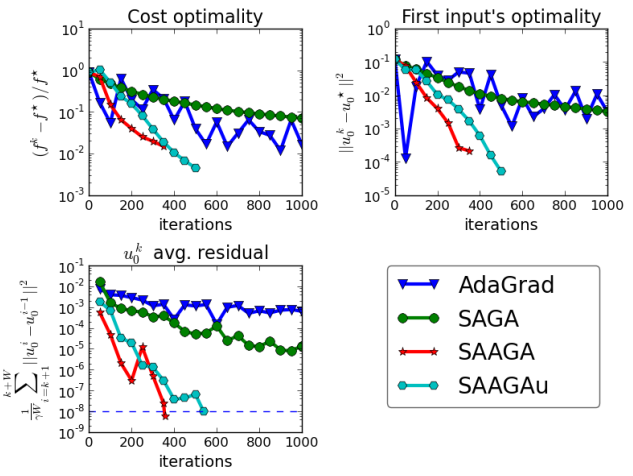


Fig. 1. Comparison of SAAGA algorithm with SAGA and ADAGRAD for SMPC problem (15) (SAAGAu refers to SAAGA algorithm with uniform sampling). The stopping criterion is as in §V-A.2 with a window size w equal to two times the number of scenarios and with a threshold of 10^{-8}. The bottom left plot shows the mean residual R_w^k, while the plots on top compare the iterates with the exact solution which was precomputed offline.

Specifically, we consider a 4 input/8 states randomly generated SMPC problem on 32 scenarios with quadratic cost and box constraints. All the data is randomly generated.

In Fig. 1 we can observe how the first input residual R_w is well correlated to the actual optimality of $u_\bullet^k$, that is, to $\|u_\bullet^k - u_\bullet^\star\|^2$, both for SAGA, SAAGA, and the uniformly sampled version of the latter, labeled SAAGAu. On the contrary, for ADAGRAD such correlation does not hold, which is due to the fact that the direction of descent $\nabla f_{[\nu]}(\underline{u}^k)$ does not provide an approximation to the full gradient. Besides, the plots show how the "hybrid" method SAAGA significantly outperforms the parents SAGA and ADAGRAD.

VII. CONCLUSIVE REMARKS

In this paper we investigated the employment of Stochastic Gradient methods for solving SMPC problems. With the goal of minimizing an expected cost f, we proposed a scenario tree based approach where f is separated among the contributions of the scenarios. The chosen decomposition *biases* the optimization in favour of the first decision variable, consistently with the receding horizon principle of MPC.

To the best of our knowledge, the analysis in the paper is the first theoretical study of the performance of (Aggregated) Stochastic Gradient methods for solving SMPC problems. The main contribution of the paper is SAAGA, an adaptive scaling of the state-of-the-art SAGA algorithm, which in practice significantly outperforms the parent scheme yet mantaining the same computational complexity per iteration, i.e., the same as SGD iterations.

Aggregated Stochastic Gradient methods have convergence rates comparable to those of full methods, yet with significantly cheaper iterations. The price is paid in terms of memory requirements, if compared to the non-aggregated counterparts. Similarly to what currently being done in [16] this motivates investigating the possibility to consider limited memory variants tailored for SMPC problems or, possibly, for separable convex optimization programs in general.

We believe that the present paper can be a starting point promoting theoretically grounded research towards the use of stochastic algorithms for efficiently solving SMPC.

REFERENCES

[1] D. Q. Mayne, "Model Predictive Control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.

[2] D. Van Hessem, C. Scherer, and O. Bosgra, "LMI-based closed-loop economic optimization of stochastic process operation under state and input constraints," in *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, vol. 5. IEEE, 2001, pp. 4228–4233.

[3] I. Batina, A. A. Stoorvogel, and S. Weiland, "Optimal control of linear, stochastic systems with state and input constraints," in *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, vol. 2. IEEE, 2002, pp. 1564–1569.

[4] M. Ono and B. C. Williams, "Iterative risk allocation: A new approach to robust model predictive control with a joint chance constraint," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*. IEEE, 2008, pp. 3427–3432.

[5] P. Patrinos, S. Trimboli, and A. Bemporad, "Stochastic MPC for real-time market-based optimal power dispatch." in *CDC-ECE*. IEEE, 2011, pp. 7111–7116.

[6] H. Heitsch and W. Römisch, "Scenario tree reduction for multistage stochastic programs," *Computational Management Science*, vol. 6, no. 2, pp. 117–133, 2009.

[7] G. C. Calafiore and L. Fagiano, "Stochastic model predictive control of lpv systems via scenario optimization," *Automatica*, vol. 49, no. 6, pp. 1861–1866, 2013.

[8] A. Sampathirao, P. Sopasakis, A. Bemporad, and P. Patrinos, "Distributed solution of stochastic optimal control problems on gpus," in *54 IEEE Conf. Decision and Control*, Osaka, Japan, Dec 2015.

[9] A. Defazio, F. R. Bach, and S. Lacoste-Julien, "SAGA: a fast incremental gradient method with support for non-strongly convex composite objectives," *CoRR*, vol. abs/1407.0202, 2014.

[10] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *The Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[11] I. Necoara, D. N. Clipici, P. Patrinos, and A. Bemporad, "MPC for power systems dispatch based on stochastic optimization," in *Proceedings of the 19th IFAC World Congress*. IFAC, August 2014, pp. 11 147–11 152.

[12] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1995.

[13] D. Blatt, A. O. Hero, and H. Gauchman, "A convergent incremental gradient method with a constant step size," *SIAM J. on Optimization*, vol. 18, no. 1, pp. 29–51, 2 2007.

[14] M. W. Schmidt, N. Le Roux, and F. Bach, "Minimizing finite sums with the Stochastic Average Gradient," *CoRR*, vol. abs/1309.2388, 2013.

[15] P. Tseng and S. Yun, "Incrementally updated gradient methods for constrained and regularized optimization," *Journal of Optimization Theory and Applications*, vol. 160, no. 3, pp. 832–853, 2014.

[16] M. Schmidt, R. Babanezhad, M. Osama Ahmed, A. Defazio, A. Clifton, and A. Sarkar, "Non-uniform Stochastic Average Gradient method for training conditional random fields," *ArXiv e-prints*, April 2015.

[17] A. Nedić, *Subgradient Methods for Convex Minimization*. Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2002.

[18] N. Cesa-Bianchi, A. Conconi, and C. Gentile, "On the generalization ability of on-line learning algorithms," *Information Theory, IEEE Transactions on*, vol. 50, no. 9, pp. 2050–2057, 2004.

[19] J. A. Primbs, "A soft constraint approach to stochastic receding horizon control," in *Decision and Control, 2007 46th IEEE Conference on*, Dec 2007, pp. 4797–4802.

[20] N. Gröwe-kuska, H. Heitsch, and W. Römisch, "Scenario reduction and scenario tree construction for power management problems," in *Power Management Problems, IEEE Bologna Power Tech Proceedings*, 2003.