Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 50-1 (2017) 11865–11870

Proximal Limited-Memory Quasi-Newton Methods for Scenario-based Stochastic Optimal Control [★]

Ajay Kumar Sampathirao [*], **Pantelis Sopasakis** [*]
Alberto Bemporad [*] and **Panagiotis Patrinos** [**]

[*] *IMT Institute for Advanced Studies Lucca, Piazza San Ponziano 6, Lucca 55100, Italy.*
[**] *KU Leuven, Department of Electrical Engineering (ESAT), STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics & Optimization in Engineering (OPTEC), Kasteelpark Arenberg 10, 3001 Leuven, Belgium.*

Abstract: Stochastic optimal control problems are typically of rather large scale involving millions of decision variables, but possess a certain structure which can be exploited by first-order methods such as forward-backward splitting and the alternating direction method of multipliers (ADMM). In this paper, we use the *forward-backward envelope*, a real-valued continuously differentiable penalty function, to recast the dual of the original nonsmooth problem as an unconstrained problem which we solve via the limited-memory BFGS algorithm. We show that the proposed method leads to a significant improvement of the convergence rate without increasing much the computational cost per iteration.

© 2017, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.

Keywords: Convex optimization; stochastic optimal control; large-scale optimization; proximal Newton methods.

1. INTRODUCTION

1.1 *Motivation and Background*

Scenario-based stochastic model predictive control is becoming increasingly popular in control applications for its ability to deliver control actions with foresight under uncertainty and has been used for the control of power dispatch (Hans et al., 2015; Patrinos et al., 2011), HVAC of buildings (Zhang et al., 2013), macroeconomic systems (Patrinos et al., 2014), supply chains (Schildbach and Morari, 2016) and many another. The involved optimization problems are typically of large dimension (involving millions of decision variables), but they possess a rich structure which gradient-based methods have been shown to be able to exploit (Sampathirao et al., 2015, 2016). Such methods converge at a rate of $\mathcal{O}(1/k)$ and $\mathcal{O}(1/k^2)$ using Nesterov's extrapolation technique (Nesterov, 2012). Nevertheless, first-order methods are sensitive to ill conditioning which may not always be possible to mitigate by preconditioning.

A straightforward approach to improve the convergence properties of first-order methods is to introduce second-order information. However, this is not available in many cases of interest, or, it is very hard to compute. The popular BFGS method produces successive approximations of the Hessian (Nocedal and Wright, 2006) and the sequence of its iterates converges Q-superlinearly to the optimal solution, but comes with a severe limitation: one needs to store and update a very large dense matrix; it is thus unsuitable for large-scale optimization.

The limited-memory BFGS (L-BFGS) method has been successfully used for the numerical solution of unconstrained problems (Liu and Nocedal, 1989) and recently also for huge-scale problems (Chen et al., 2014). It implicitly updates a diagonal approximation of the Hessian using a computationally cheap algorithm known as the *two-loop recursion* (Nocedal and Wright, 2006). Despite its popularity it comes with two limitations which have hindered its use for the solution of optimal control problems. First, it can only be applied to unconstrained optimal control problems or problems with only box constraints on the input variables (Byrd et al., 1995) and second, it cannot be applied to nonsmooth problems.

These limitations are lifted using the *forward-backward envelope* (FBE) of the original optimization problem which allows us to reformulate it as an unconstrained problem of a continuously differentiable function (Patrinos et al., 2014; Themelis et al., 2016b,a). In this paper we show that the application of the L-BFGS method to the FBE leads to a noticeable improvement of the convergence speed without a significant increase in the computational cost per iteration.

1.2 *Contribution*

We previously showed that stochastic optimal control problems possess a certain structure which can be ex-

[★] The work of the second author was supported by the EU-funded H2020 research project DISIRE, grant agreement No. 636834. The work of the fourth author was supported by the KU Leuven Research Council under BOF/STG-15-043.

ploited for their efficient numerical solution using an accelerated proximal gradient (APG) algorithm (Sampathirao et al., 2015). In this paper, we formulate the Fenchel dual optimization problem introducing a splitting which has favourable separability properties. We employ a quasi-Newtonian algorithm combining the limited-memory BFGS method with the forward-backward envelope function to achieve faster convergence. The proposed method involves only matrix-vector products and enables an implementation where operations across all nodes of the scenario tree at every stage are executed in parallel.

1.3 Notation

Let $\mathbb{R}$, $\mathbb{N}$, $\mathbb{R}^n$, $\mathbb{R}^{m\times n}$, $\mathbb{S}^n_+$, $\mathbb{S}^n_{++}$ denote the sets of real numbers, nonnegative integers, column real vectors of length n, real matrices of dimensions m-by-n, symmetric positive semidefinite and positive definite n-by-n matrices respectively. Let $\overline{\mathbb{R}} = \mathbb{R}\cup\{\pm\infty\}$ denote the set of extended real numbers. The transpose of a matrix A is denoted by $A^\top$ and $\langle x, y\rangle$ stands for the standard inner product of x and y. The set of of nonnegative integers $\{k_1, k_1+1,\ldots,k_2\}, k_2 \geq k_1$ is denoted by $\mathbb{N}_{[k_1,k_2]}$.

The *indicator function* of a set $C \subseteq \mathbb{R}^n$ is the extended-real valued function $\delta(\cdot|C) : \mathbb{R}^n \to \overline{\mathbb{R}}$ and for $x \in C$ it is $\delta(x|C) = 0$ and $\delta(x|C) = +\infty$ otherwise. A function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ is called *lower semi-continuous* or *closed* if for every $x \in \mathbb{R}^n$, $f(x) = \liminf_{z\to x} f(z)$. A function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ is called *proper* if there is an $x \in \mathbb{R}^n$ so that $f(x) < \infty$ and $f(x) > -\infty$ for all $x \in \mathbb{R}^n$. For a closed convex function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$, we define its *conjugate* $f^* : \mathbb{R}^n \to \overline{\mathbb{R}}$ as $f^*(x^*) = \sup_x\{\langle x, x^*\rangle - f(x)\}$. A mapping $F : \mathbb{R}^n \to \mathbb{R}^m$ is called *β-Lipschitz continuous*, with $\beta \geq 0$, if $\|F(x_1) - F(x_2)\|_* \leq \beta\|x_1 - x_2\|$ for every $x_1, x_2 \in \mathbb{R}^n$. We call f *σ-strongly convex* if $f(x) - \frac{\sigma}{2}\|x\|_2^2$ is a convex function. Unless otherwise stated $\|\cdot\|$ stands for $\|\cdot\|_2$.

Every nonempty closed convex set $C \subseteq \mathbb{R}^n$ defines the convex function $\operatorname{proj}(x|C) = \operatorname{argmin}_{c\in C}\|x - c\|_2$, which is called the *(Euclidean) projection* of x onto C. The Euclidean distance of an $x \in \mathbb{R}^n$ from C is defined as $\operatorname{dist}(x|C) = \min_{c\in C}\|x - c\|_2$.

2. PROBLEM STATEMENT

2.1 Stochastic optimal control

We first provide a formal statement of general stochastic optimal control problems for linear dynamical systems. Let $(\Omega, \mathscr{F}, \mathrm{P})$ be a probability space and $\{\varnothing, \Omega\} = \mathscr{F}_0 \subseteq \mathscr{F}_1 \subseteq \ldots \subseteq \mathscr{F}_{N-1} = \mathscr{F}$ be a nested sequence of σ-algebras known as a *filtration* (Shapiro et al., 2009). We shall use the notation $v \lhd \mathscr{F}_k$ to denote that $v : \Omega \to \mathbb{R}$ is a $\mathscr{F}_k$-measurable random variable — this essentially means that v depends only on information that is available up to time k. Consider the stochastic discrete-time linear system

$$x_{k+1} = A_{\xi_k}x_k + B_{\xi_k}u_k + w_{\xi_k}, \tag{1}$$

where $\xi_k \lhd \mathscr{F}_k$, $u_k \lhd \mathscr{F}_{k-1}$ and with known initial condition $x_0 = p$. This practically means that u_k is a *causal* control law, i.e., it is a function $u_k = \psi_k(p, \xi_0, \ldots, \xi_{k-1})$ for $k \in \mathbb{N}_{[1,N-1]}$ and $u_0 = \psi_0(x_0)$[1].

[1] In some applications we may assume that $u_k \lhd \mathscr{F}_k$, i.e., u_k is decided as a function of p and all $\xi_0, \ldots, \xi_k$.

A *stochastic optimal control problem* for (1) with horizon N and decision variable $\pi = \{u_k\}_{k\in\mathbb{N}_{[0,N-1]}}$ can be formulated as

$$V^\star(p) = \min_\pi \; \mathbb{E}\left[V_f(x_N, \xi_N) + \sum_{k=0}^{N-1}\ell_k(x_k, u_k, \xi_k)\right], \tag{2}$$

subject to (1) and the condition $x_0 = p$ with $\ell_k \lhd \mathscr{F}_k$, $V_f \lhd \mathscr{F}_{N-1}$ and $\mathbb{E}$ is the expectation operator of the product probability space of the filtered probability space $(\Omega, \mathscr{F}, \{\mathscr{F}_k\}_k, \mathrm{P})$. In (2) functions ℓ_k and V_f are extended-real-valued functions which, as we are about to discuss, can be used to encode hard and/or soft constraints, so this formulation is quite general (Sampathirao et al., 2016, 2015).

We assume that in (2) the cost functions ℓ_k are written as $\ell_k(x, u, \xi) = \phi_k(x, u, \xi) + \bar\phi_k(F_kx + G_ku, \xi)$, with F_k, G_k are functions of ξ (thus $F_k, G_k \lhd \mathscr{F}_k$), where ϕ_k is real-valued, smooth in (x, u), while $\bar\phi_k$ is an extended-real-valued function, lower semi-continuous, proper, convex and possibly nonsmooth. Likewise, V_f can be decomposed as $V_f(x, \xi) = \phi_N(x, \xi) + \bar\phi_N(F_Nx, \xi)$.

As an example, we may use $\bar\phi_k$ to encode arbitrary hard constraints on states and inputs of the form $F_kx_k + G_ku_k \in Y_k$ by choosing

$$\bar\phi_k(\cdot) = \delta(\cdot|Y_k), \tag{3}$$

where Y_k are nonempty convex closed sets for which projections $\operatorname{proj}(\cdot|Y_k)$ can be easily computed. Soft constraints can be encoded by choosing

$$\bar\phi_k(\cdot) = \eta_k \operatorname{dist}(\cdot|Y_k), \tag{4}$$

where $\eta_k > 0$.

The smooth part of the stage cost ℓ_k is a quadratic function of the form

$$\phi_k(x, u, \xi) = \begin{bmatrix}x\\u\end{bmatrix}^\top \begin{bmatrix}Q_k & S_k^\top\\ S_k & R_k\end{bmatrix}\begin{bmatrix}x\\u\end{bmatrix} + q_k^\top x + r_k^\top u, \tag{5}$$

where $R_k \in \mathbb{S}^{n_u}_{++}$, $Q_k \in \mathbb{S}^{n_x}_+$, $\left[\begin{smallmatrix}Q_k & S_k^\top\\ S_k & R_k\end{smallmatrix}\right] \in \mathbb{S}^{n_x}_+$ and R_k, Q_k, S_k, q_k, r_k may depend on ξ. The smooth part of the terminal cost function V_f is a quadratic function $\phi_N(x, \xi) = x^\top P_Nx + p_N^\top x$, with $P_N \in \mathbb{S}^{n_x}_{++}$ and P_N, p_N may depend on ξ. The function $\bar\phi_N$ can be selected in the same way as we have explained for $\bar\phi_k$, e.g., terminal constraints of the form $F_Nx_N \in X_f$ can be encoded using $\bar\phi_N(\cdot) = \delta(\cdot|X_f)$, where X_f is assumed to be such that $\operatorname{proj}(\cdot|X_f)$ can be easily evaluated computationally.

2.2 Scenario-based formulation

The scenario-based formulation of (2) accrues from the assumption that $\mathscr{F}_{N-1}$ is finite and produces the scenario tree structure shown in Fig. 1. A scenario tree describes the probable evolution of the state sequence $\{x_k\}_{k\in\mathbb{N}_{[0,N]}}$. The elementary events $\{\xi_N^i\}_{i\in\mathbb{N}_{[1,\mu]}}$ identify a set of *final outcomes* which correspond to the *leaf nodes* of the scenario tree. In turn, each leaf node identifies a single scenario, i.e., a sequence of realizations of the random process $\{\xi_k\}_{k\in\mathbb{N}_{[0,N]}}$. The tree is partitioned in N stages. The observable scenarios at stage k are the *nodes* of the tree at that stage; the number of nodes at stage k is denoted by

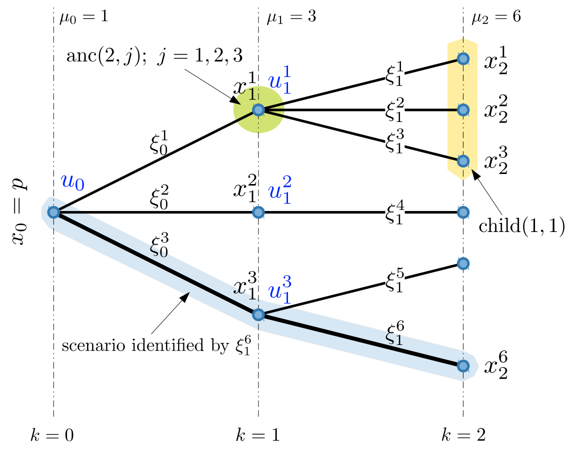

Fig. 1. Scenario tree structure describing the evolution of the state.

μ_k. The probability that at stage k, the scenario ξ_N^i occurs is denoted by p_k^i.

As shown in Fig. 1, at stage $k \in \mathbb{N}_{[0,N-1]}$ the i-th node defines a set of *children nodes* at stage $k+1$ denoted by $\mathrm{child}(k,i) \subseteq \mathbb{N}_{[1,\mu_{k+1}]}$. Every node at stage $k \in \mathbb{N}_{[1,N]}$ has a unique ancestor node at stage $k-1$ denoted by $\mathrm{anc}(k,i) \in \mathbb{N}_{[1,\mu_{k-1}]}$.

The system dynamics along scenarios can be written as

$$x_{k+1}^j = A_k^j x_k^i + B_k^j u_k^i + w_k^j, \tag{6}$$

with $i = \mathrm{anc}(k+1,j)$, $A_k^j = A_{\xi_k^j}$, $B_k^j = B_{\xi_k^j}$ and $w_k^j = w_{\xi_k^j}$.

Let x be a vector comprising all x_k^i and u_k^i and let $\mathcal{Z}(p)$ be the linear space of all x satisfying (6) with $x_0 = p$. Define

$$f(x) = \sum_{k=0}^{N-1}\sum_{i=1}^{\mu_k} p_k^i \phi_k(x_k^i, u_k^i, \xi_k^i) + \sum_{i=1}^{\mu_N} p_{N-1}^i \phi_N(x_N^i, \xi_N^i) + \delta(x|\mathcal{Z}(p)), \tag{7a}$$

$$g(Hx) = \sum_{k=0}^{N-1}\sum_{i=1}^{\mu_k} p_k^i \bar{\phi}_k^i(F_k^i x_k^i + G_k^i u_k^i, \xi_k^i) + \sum_{i=1}^{\mu_N} p_{N-1}^i \bar{\phi}_N^i(F_N^i x_N^i, \xi_N^i). \tag{7b}$$

Given that ϕ_k are given as in (5), function f is strongly convex, therefore f^* is differentiable with L-Lipschitz gradient because of (Rockafellar and Wets, 1998, Prop. 12.60). Now problem (2) can be written as

$$P^\star = \min_x f(x) + g(Hx), \tag{8}$$

where H is a linear operator with $z = Hx$ with $z_k^i = F_k^i x_k^i + G_k^i u_k^i$ for $k \in \mathbb{N}_{[0,N-1]}$, $i \in \mathbb{N}_{[0,\mu_k]}$ and $z_N^i = F_N^i x_N^i$. The Fenchel dual of (8) is

$$D^\star = \min_y f^*(-H^\top y) + g^*(y). \tag{9}$$

Strong duality holds for the above problem, i.e., $P^\star = D^\star$, under weak assumptions on the domains of $\bar{\phi}_k$.

For notational convenience we define $f_\circ(y) := f^*(-H^\top y)$, thus $\nabla f_\circ(y) = -H\nabla f^*(-H^\top y)$.

3. NUMERICAL ALGORITHM

3.1 The Forward-Backward Envelope Function

The proximal operator of a proper, closed, convex function g plays a major role in modern optimization theory and is defined as

$$\mathrm{prox}_{\lambda g}(v) = \mathrm{argmin}_z\{g(z) + \tfrac{1}{2\lambda}\|v - z\|^2\}. \tag{10}$$

Proximal operators of a great variety of functions including indicators of sets, distance-to-set functions and norms can be easily evaluated analytically and at a very low computational cost (Combettes and Pesquet, 2010). For example, the proximal operator of ϕ_k in (3) is the projection on Y_k, that is $\mathrm{prox}_{\gamma\delta(\cdot|Y_k)}(v) = \mathrm{proj}(v \mid Y_k)$.

A simple optimality condition for (9) is

$$y - \mathrm{prox}_{\lambda g^*}(y - \lambda \nabla f_\circ(y)) = 0, \tag{11}$$

for some $\lambda > 0$ (Parikh and Boyd, 2013). By virtue of the Moreau decomposition formula, (11) is equivalently written as

$$\nabla f_\circ(y) + \mathrm{prox}_{\lambda^{-1}g}(\lambda^{-1}y - \nabla f_0(y)) = 0. \tag{12}$$

We define the *forward-backward mapping*

$$T_\lambda(y) := \mathrm{prox}_{\lambda g^*}(y - \lambda\nabla f_\circ(y)), \tag{13}$$

which, using the Moreau decomposition property, becomes

$$T_\lambda(y) = y - \lambda\nabla f_\circ(y) - \lambda\,\mathrm{prox}_{\lambda^{-1}g}(\lambda^{-1}y - \nabla f_\circ(y)), \tag{14}$$

and we also define the *fixed-point residual* mapping

$$R_\lambda(y) := \lambda^{-1}(y - T_\lambda(y)). \tag{15}$$

The aforementioned optimality condition (11) is equivalently written as $R_\lambda(y) = 0$, that is, solving the dual optimization problem (9) becomes equivalent to finding a zero of the operator R_λ.

The *forward-backward envelope* (FBE) of (9) is a real-valued function φ_λ given by (Patrinos et al., 2014; Patrinos and Bemporad, 2013)

$$\varphi_\lambda(y) = f_\circ(y) + g^*(T_\lambda(y)) - \lambda\langle\nabla f_\circ(y), R_\lambda(y)\rangle + \tfrac{\lambda}{2}\|R_\lambda(y)\|^2,$$

and, provided that $f_\circ$ is twice continuously differentiable, φ_λ is continuously differentiable with Lipschitz-continuous gradient given by

$$\nabla\varphi_\lambda(y) = (I - \lambda\nabla^2 f_\circ(y))R_\lambda(y). \tag{16}$$

Also, since $f_\circ$ is convex quadratic, φ_λ is also convex (Patrinos et al., 2014).

The most important property of the FBE is that for $\lambda \in (0, 1/L)$, the set of minimizers of (9) coincides with

$$\begin{aligned}\mathrm{argmin}\,\varphi_\lambda &\equiv \mathrm{zer}\,\nabla\varphi_\lambda := \{y : \nabla\varphi_\lambda(y) = 0\}\\ &= \mathrm{argmin}\, f_\circ(y) + g^*(y) = \mathrm{zer}\,R_\lambda.\end{aligned}$$

Essentially, the problem of solving the dual optimization problem (9) is equivalent to the unconstrained minimization of the continuously differentiable function φ_λ which is in turn equivalent to finding a zero of the fixed-point residual operator.

We should highlight here that the value and gradient of the FBE are computed at the computational cost of a forward-backward step. Moreover, for the evaluation of $\nabla\varphi_\lambda(y)$ it suffices to have a way to compute products $\nabla^2 f^*(y)\cdot d$. If a closed-form formula is not available it can be evaluated numerically.

Overall, the proposed algorithmic scheme assumes the availability of an *oracle* which allows us to compute the dual gradient $\nabla f^*(-H^\top y)$ at a given point y and Hessian-vector products $\nabla^2 f_\circ(x)\cdot d$ at given points x and d. The complexity of the algorithm can then be evaluated on the basis of these oracle invocations.

3.2 Computation of the dual gradient

The efficient computation of the dual gradient is of crucial importance for the performance of the algorithm we are about to describe. By virtue of the Conjugate Subgradient Theorem (Rockafellar, 1976), we have that

$$\nabla f^*(-H^\top y) = \operatorname{argmin}_z\{\langle z, H^\top y\rangle + f(z)\}. \tag{17}$$

Since f is given by (7a), (17) can be solved by dynamic programming using Algorithm 1 (see Sampathirao et al. (2015)), where Φ_k^i, Θ_k^i, D_k^i, Λ_k^i, K_k^i and σ_k^i, c_k^i are computed once offline following the Ricatti-type recursion of (Sampathirao et al., 2015, Algorithm 1).

Algorithm 1 Dual gradient computation

```
$q_N^i \leftarrow y_N^i + p_N^i$, for all $i \in \mathbb{N}_{[1,\mu_N]}$
for $k = N-1,\dots,0$ and $i = 1,\dots,\mu_k$ do in parallel
    $u_k^i \leftarrow \Phi_k^i y_k^i + \sigma_k^i$
    $q_k^i \leftarrow D_k^{i\prime} y_k^i + c_k^i$
for $k = N-1,\dots,0$ do
    for $i = 1,\dots,\mu_k$ do in parallel
        $u_k^i \leftarrow \sum_{j\in\mathrm{child}(k,i)} \Theta_k^j q_{k+1}^j$
        $q_k^i \leftarrow \sum_{j\in\mathrm{child}(k,i)} \Lambda_k^{j\prime} q_{k+1}^j$
$x_0^1 = p$
for $k = 0,\dots,N-1$ do
    for $i = 1,\dots,\mu_k$ do in parallel
        $u_k^i \leftarrow K_k^i x_k^i + u_k^i$
        for $j \in \mathrm{child}(k,i)$ do in parallel
            $x_{k+1}^j \leftarrow A_k^j x_k^i + B_k^j u_k^i + w_k^j$
```

For a q-ary tree and assuming that $F_k^i \in \mathbb{R}^{n_c\times n_x}$, Algorithm 1 involves $\mu[(2n_c-1)(n_x+n_u) + n_u(2n_x+1) + n_x + q(n_x+n_u)(2n_x+1)]$ flops where $\mu = \sum_{k=0}^{N-1}\mu_k$. The complexity of the computation of the dual gradient is therefore linear in the prediction horizon and also linear in the total number of nodes of the scenario tree.

3.3 Computation of the dual Hessian

The computation of $\nabla\varphi_\lambda(y)$ requires the computation of products of the form $\nabla^2 f_\circ\cdot d$. Notice that to a great extent the computations in Algorithm 2 can be parallelized. The dual Hessian is then used for the computation of $\nabla\varphi_\lambda$.

Again assuming that $F_k^i \in \mathbb{R}^{n_c\times n_x}$, the total flop count for Algorithm 2 tallies up to $\mu[(2n_c-1)(n_u+2n_x)+4n_xn_u+2qn_x(n_x+n_u-1)]$, which is of the same order of magnitude as the cost of Algorithm 1.

3.4 Computation of $\nabla\varphi_\lambda$

The gradient of the FBE, $\nabla\varphi_\lambda(y)$, can be computed as in (16) where $R_\lambda(y)$ is computed as

$$R_\lambda(y) = \lambda^{-1}(z(y) - Hx(y)), \tag{18a}$$

Algorithm 2 Computation of Hessian-vector products

```
Require: Vector $d$
$\hat{q}_N^i \leftarrow d_N^i, \forall i \in \mathbb{N}_{[1,\mu_N]}$
for $k = N-1,\dots,0$ do
    for $i = 1,\dots,\mu_k$ do in parallel
        $\hat{u}_k^i \leftarrow \Phi_k^i d_k^i + \sum_{j\in\mathrm{child}(k,i)} \Theta_k^j \hat{q}_{k+1}^j$
        $\hat{q}_k^i \leftarrow D_k^{i\top} d_k^i + \sum_{j\in\mathrm{child}(k,i)} \Lambda_k^{j\top} \hat{q}_{k+1}^j$
$\hat{x}_0^1 = 0$
for $k = 0,\dots,N-1$ do
    for $i = 1,\dots,\mu_k$ do in parallel
        $u_k^i \leftarrow K_k^i \hat{x}_k^i + \hat{u}_k^i$
        for $j \in \mathrm{child}(k,i)$ do in parallel
            $\hat{x}_{k+1}^j \leftarrow A_k^j \hat{x}_k^i + B_k^j \hat{u}_k^i$
```

where

$$x(y) = \operatorname{argmin}_z\{\langle z, H^\top y\rangle + f(z)\}, \tag{18b}$$

$$z(y) = \operatorname{prox}_{\lambda^{-1}g}\{\lambda^{-1}y + Hx(y)\}, \tag{18c}$$

where $x(y) = \nabla f^*(-H^\top y)$ is computed by Algorithm 1 and $z(y)$ is a proximal step. The latter typically consists in simple element-wise operations which can be fully parallelized.

3.5 L-BFGS method for the FBE

Algorithm 3 summarizes the basic steps of the proposed solution. At every iteration, an L-BFGS direction d^ν is computed using the two-loop recursion of (Nocedal and Wright, 2006, Algorithm 7.4), that is, in line 3 of Algorithm 3 the matrix B^ν — which is an approximation of the inverse Hessian when this exists — does not need to be constructed or stored. The computation of d^ν requires only $4mn_d$ multiplications, where m is the memory length of the LBFGS buffer and n_d is the dimension of d^ν. This step involves the computation of the gradient of the FBE at y^ν which is performed as discussed in Section 3.4.

The dual vector y^ν is updated as in line 4 where τ_ν is chosen so as to satisfy the Wolfe conditions (Nocedal and Wright, 2006, Sec. 3.1):

$$\varphi_\lambda(y^{\nu+1}) \le \varphi_\lambda(y^\nu) + c_1\tau_\nu\langle\nabla\varphi_\lambda(y^\nu), d^\nu\rangle \tag{19a}$$

$$\langle\nabla\varphi_\lambda(y^{\nu+1}), d^\nu\rangle \ge c_2\langle\nabla\varphi_\lambda(y^\nu), d^\nu\rangle, \tag{19b}$$

where $0 < c_1 < c_2 < 1$. Here we used $c_1 = 10^{-4}$ and $c_2 = 0.9$. The first inequality is a sufficient decrease condition, while the second one is known as the *curvature condition* and is used to rule out unacceptably short step lengths. The existence of intervals of τ_ν which satisfy the Wolfe conditions is guaranteed and such values can be determined with the line search method proposed in (Nocedal and Wright, 2006).

Typically, in quasi-Newton methods for the Hessian approximations to be positive definite, the Wolfe conditions are used to determine the step-size τ_ν and an inexact line search is used to compute an appropriate step size as in (Nocedal and Wright, 2006, Algorithm 3.5).

Although not necessary, quasi-Newton methods benefit from preconditioning. A simple preconditioning which aims at eliminating the effect of the probabilities p_k^i is employed here: the dual variables are scaled by dividing each y_k^i by $\sqrt{p_k^i}$ where $p_k^i > 0$.

Algorithm 3 Forward-Backward L-BFGS

```
Require: $\lambda \in (0, 1/L)$, $y^0$, $m$ (memory), $\epsilon$ (tolerance),
  $\nu_{\max}$ (maximum iterations)
  Initialize an LBFGS buffer of length $m$
  while $\|R_\lambda(y^\nu)\| > \epsilon \|R_\lambda(y^0)\|$ and $\nu \leq \nu_{\max}$ do
      $d^\nu \leftarrow -B^\nu \nabla\varphi_\lambda(y^\nu)$ (LBFGS direction)
      $y^{\nu+1} \leftarrow y^\nu + \tau_\nu d^\nu$ where $\tau_k$ satisfies (19)
      $s^\nu \leftarrow y^{\nu+1} - y^\nu$, $q^\nu \leftarrow \nabla\varphi_\lambda(y^{\nu+1}) - \nabla\varphi_\lambda(y^\nu)$
      $\rho_\nu \leftarrow 1/\langle s^\nu, q^\nu\rangle$
      Push $(s^\nu, q^\nu, \rho_\nu)$ in the LBFGS buffer
      $\nu \leftarrow \nu + 1$
```

Algorithm 3 is terminated once the fixed-point residual becomes adequately small; we use the termination condition $\|R_\lambda(y^\nu)\| \leq \epsilon\|R_\lambda(y^0)\|$. The algorithm produces a sequence y^ν for which $\liminf_\nu \|R_\lambda(y^\nu)\| = 0$, therefore the termination condition will be satisfied within finitely many iterations for any ϵ (Powell, 1976).

3.6 Nonlinear Conjugate Gradient

The use of the forward-backward envelope allows the use of other smooth optimization methods such as the *Polak-Ribière+ nonlinear conjugate gradient* (PR+) method (Nocedal and Wright, 2006; Grippo and Lucidi, 1997). At every iteration we compute

$$\beta^{\mathrm{PR}}_{\nu+1} = \frac{\nabla\varphi_\lambda(y^{\nu+1})^\top(\nabla\varphi_\lambda(y^{\nu+1}) - \nabla\varphi_\lambda(y^\nu))}{\|\nabla\varphi_\lambda(y^\nu)\|^2}, \qquad (20)$$

and $\beta^{\mathrm{PR}}_0 = 0$ and use the direction $d^{\nu+1} = -\nabla\varphi_\lambda(y^{\nu+1}) + \beta^{\mathrm{PR+}}_{\nu+1} d^\nu$, where $\beta^{\mathrm{PR+}}_{\nu+1} = \max\{0, \beta^{\mathrm{PR}}_{\nu+1}\}$. In PR+, τ_ν is chosen so as to satisfy the *strong Wolfe conditions* where (19b) is replaced by $|\langle\nabla\varphi_\lambda(y^{\nu+1}), d^\nu\rangle| \leq c_2|\langle\nabla\varphi_\lambda(y^\nu), d^\nu\rangle|$ together with the descent condition $\langle\nabla\varphi_\lambda(x_{k+1}), d_{k+1}\rangle \leq -c_3\|\nabla\varphi_\lambda(x_{k+1})\|^2$ (Gilbert and Nocedal, 1992). Hereafter we use the values $c_1 = 10^{-4}$, $c_2 = 0.9$ and $c_3 = 10^{-4}$.

However, as we will discuss in the following section, although PR+ leads to a significant reduction in the number of iterations compared to APG, it does not seem to be faster than it due to the additional cost required in the line search.

4. SIMULATIONS

We formulated the stochastic optimal control problem for a linear system with additive and multiplicative uncertainty as in (1) to evaluate the proposed algorithm. The linear system we have considered is the spring-mass set-up as in (Sampathirao et al., 2015). This system has $m = 5$ masses weighing 2 kg connected with $m - 1$ linear spring-dampers with stiffness constant $\kappa = 1\,\mathrm{N/m}$ and damping ratio $\beta = 0.1$. The manipulated variables are the forces we may exercise on each spring along their principal axes and the state variables are the positions and speeds of the masses, i.e., it has $2m$ state variables and $m-1$ input variables. We assume that the system dynamics is obtained by discretizing the continuous-time dynamics with sampling time $T = 0.5\,\mathrm{s}$. On the system state and input variables we impose the constraints $-4 \leq x^i_k \leq 4$ and $-2 \leq u^i_k \leq 2$ for all $k \in \mathbb{N}_{[0,1]}$ and $i \in \mathbb{N}_{[1,\mu_k]}$. The stage cost was chosen to be $\ell(x, u, \xi) = x'Qx + u'Ru$ with $Q = I_{n_x}$ and $R = I_{n_u}$.

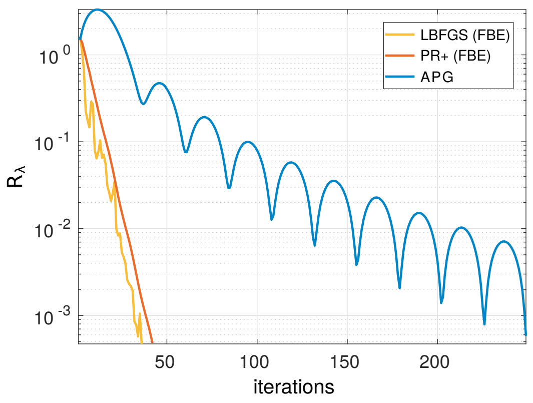

Fig. 2. Convergence of LBFGS, PR+ and APG for a binary tree with 1024 scenarios.

We consider scenario trees whose numbers of scenarios are powers of 2 from 2 to 2^{10}. All trees are taken with a fixed horizon $N = 10$ and in their first stages are binary, i.e., with a branching factor 2 and eventually evolve without branching until the end of the horizon. We consider a buffer size with memory 5 for the LBFGS-FBE algorithm. The convergence condition for all algorithms is $\epsilon = 5\cdot10^{-4}$. We generate 100 random initial states x_0 for the stochastic optimal control problem. All algorithms are implemented in MATLAB and executed on a $4 \times 2.60\,\mathrm{GHz}$ Intel i5 machine with 8 GB RAM running 64-bit Ubuntu 14.04.

Although L-BFGS incurs a high cost per iteration (approximately double) compared to APG, as we may observe in Fig. 2 it converges much faster. Overall, L-BFGS requires on average a significantly lower number of floating point operations (flop) as Fig. 3 shows. For example, for the case of 1024 scenarios, LBFGS requires 191 Mflop on average (max 742 Mflop) compared to 512 Mflop for APG (max 1.743 Gflop) and 463 Mflop for PR+ (max 3.035 Gflop).

In Fig. 4 we see that PR+ outperforms AGP in terms of the total number of iterations required for convergence with $\epsilon = 5 \cdot 10^{-4}$. However, PR+ overall requires almost as many oracle calls as APG which compromises the advantages of its good convergence properties.

5. CONCLUSIONS

We have already shown that APG can take advantage of the problem structure and parallelize the operations involved in the computation of the dual gradient across all scenarios at each stage of the tree. As a result stochastic optimal control problems can be solved very efficiently on GPUs (Sampathirao et al., 2015, 2016). In this paper, we have demonstrated that it is possible to achieve better results using the L-BFGS method on a smooth merit function of the original optimization problem: the forward-backward envelope. By using the FBE enables the use of any other smooth optimization method such as nonlinear conjugate gradient methods. The proposed LBFGS method is superior to APG both in terms of oracle invocations and number of floating point operations. Future work will focus on the solution of stochastic optimal control problems on GPUs using the proposed method.

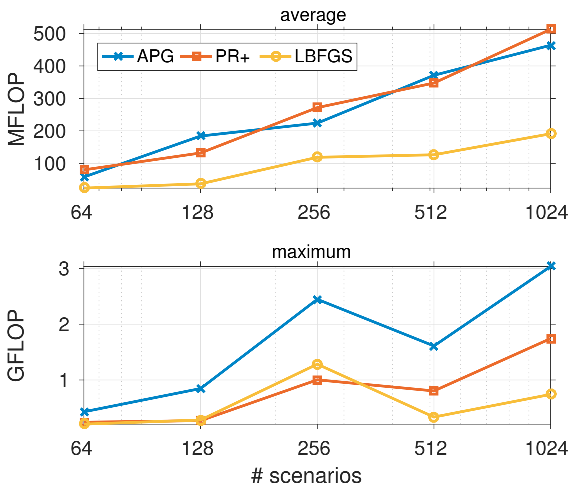

Fig. 3. Average and maximum number of floating point operations required by the proposed method compared to APG and PR+.

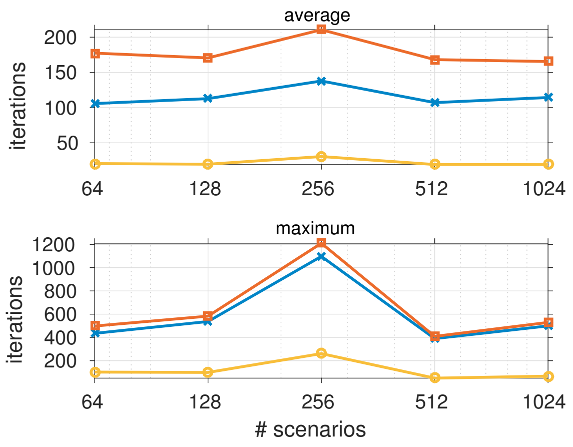

Fig. 4. Iterations required for the proposed algorithm to converge. Comparison with PR+ and APG.

REFERENCES

R.H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.*, 16(5):1190–1208, sep 1995.

W. Chen, Z. Wang, and J. Zhou. Large-scale L-BFGS using MapReduce. In *Advances in Neural Information Processing Systems*, volume 27, pages 1332–1340. Curran Associates Inc., 2014.

P.L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. Technical report, 2010. URL `http://arxiv.org/abs/0912.3522v4`.

J.C. Gilbert and J. Nocedal. Global convergence properties of conjugate gradient methods for optimization. *SIAM J. Optimization*, 2(1):21–42, 1992.

L. Grippo and S. Lucidi. A globally convergent version of the Polak-Ribière conjugate gradient method. *Mathematical Programming*, 78(3):375–391, 1997.

C.A. Hans, P. Sopasakis, A. Bemporad, J. Raisch, and C. Reincke-Collon. Scenario-based model predictive operation control of islanded microgrids. In *54 IEEE Conf. Decision and Control*, Osaka, Japan, Dec 2015.

D.C. Liu and J. Nocedal. On the limited memory BFGS method for large-scale optimization. *Mathematical Programming*, 56(1–3):503–528, 1989.

Y. Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, Dec 2012.

J. Nocedal and S.J. Wright. *Numerical optimization.* Springer, USA, 2nd edition, 2006.

N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):123–231, 2013.

P. Patrinos and A. Bemporad. Proximal Newton methods for convex composite optimization. In *IEEE CDC*, pages 2358–2363, Florence, Italy, 2013.

P. Patrinos, S. Trimboli, and A. Bemporad. Stochastic MPC for real-time market-based optimal power dispatch. In *50th IEEE CDC & ECC*, pages 7111–7116, Dec 2011.

P. Patrinos, P. Sopasakis, H. Sarmiveis, and A. Bemporad. Stochastic model predictive control for constrained discrete-time Markovian switching systems. *Automatica*, 50(10):2504–2514, October 2014.

P. Patrinos, L. Stella, and A. Bemporad. Forward-backward truncated Newton methods for convex composite optimization. Technical report, 2014. URL `http://arxiv.org/abs/1402.6655`.

M.J.D. Powell. Some global convergence properties of a variable metric algorithm for minimization without exact line searches. In *Nonlinear Programming, SIAM-AMS Proceedings*, number 9, 1976.

R. Rockafellar and R. Wets. *Variational analysis*, volume 317 of *Grundlehren der Mathematischen Wissenschaften.* Springer-Verlag, Berlin, 1998.

R.T. Rockafellar. *Convex Analysis.* Princeton university press, 1976.

A.K. 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.

A.K. Sampathirao, P. Sopasakis, A. Bemporad, and P. Patrinos. GPU-accelerated stochastic predictive control of drinking water networks. Technical report, 2016. URL `http://arxiv.org/abs/1604.01074`.

G. Schildbach and M. Morari. Scenario-based model predictive control for multi-echelon supply chain management. *European Journal of Operational Research*, 252 (2):540 – 549, 2016. ISSN 0377-2217.

A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on Stochastic Programming: Modeling and Theory.* SIAM and MPS editions, 2009.

A. Themelis, L. Stella, and P. Patrinos. Forward-backward envelope for the sum of two nonconvex functions: Further properties and nonmonotone line-search algorithms. Technical report, 2016a. URL `http://arxiv.org/abs/1606.06256`.

A. Themelis, L. Stella, and P. Patrinos. Forward-backward quasi-Newton methods for nonsmooth optimization problems. Technical report, 2016b. URL `http://arxiv.org/abs/1604.08096`.

X. Zhang, G. Schildbach, D. Sturzenegger, and M. Morari. Scenario-based MPC for energy-efficient building climate control under weather and occupancy uncertainty. In *IEEE ECC*, pages 1029–1034, July 2013.