A Fast Smoothing Procedure for Large-Scale Stochastic Programming

Martin Biel, Vien V. Mai, Mikael Johansson

Abstract— **We develop a fast smoothing procedure for solving linear two-stage stochastic programs, which outperforms the well-known L-shaped algorithm on large-scale benchmarks. We derive problem-dependent bounds for the effect of smoothing and characterize the convergence rate of the proposed algorithm. The theory suggests that the smoothing scheme can be sped up by sacrificing accuracy in the final solution. To obtain an efficient and effective method, we suggest a hybrid solution that combines the speed of the smoothing scheme with the accuracy of the L-shaped algorithm. We benchmark a parallel implementation of the smoothing scheme against an efficient parallelized L-shaped algorithm on three large-scale stochastic programs, in a distributed environment with 32 worker cores. The smoothing scheme reduces the solution time by up to an order of magnitude compared to L-shaped.**

I. Introduction

Stochastic programming [1] is a well-established approach to decision-making under uncertainty with countless applications in the control and decision sciences, including power systems [2], process control [3] and air traffic management [4]. In this paper, we focus on two-stage linear stochastic programs with fixed recourse of the form

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & c^Tx + \mathbb{E}_\xi[Q(x,\xi(\omega))] \\ \text{subject to} \quad & Ax = b, \\ & x \geq 0 \end{aligned} \tag{1}$$

where

$$\begin{aligned} Q(x,\xi(\omega)) = \min_{y\in\mathbb{R}^m} \quad & q_\xi^T y \\ \text{s.t.} \quad & T_\xi x + Wy = h_\xi \\ & y \geq 0. \end{aligned}$$

This formulation aims at determining the present decision x that minimizes the immediate cost c^Tx plus the expected future cost over a set of possible scenarios. These scenarios correspond to outcomes of a random variable ξ. Modern industrial applications of stochastic programming often result in optimization problems of a large scale. For example, the unit-commitment problem in [5] has 16 384 scenarios and four billion variables, which calls for effective parallelization procedures. This typically involves exploiting the structure of the stochastic program by mathematical decomposition.

A classical decomposition procedure for stochastic programs is the L-shaped algorithm [6], a cutting-plane method with finite convergence guarantees. The method has been heavily studied [7]–[10], and is typically able to converge to an optimal solution in few iterations. We have previously demonstrated the efficacy of this method in distributed settings [11]. We also noted that distributed L-shaped methods suffer from scalability issues due to load imbalance between master and worker nodes as the algorithm progresses. The load imbalance stems from the master problem increasing in size as cutting planes are added. In contrast, subgradient methods such as the quasigradient algorithm [12] perform a projection step each iteration, which involves solving a quadratic program that does not increase in size. Therefore, a parallel subgradient method with low iteration complexity could potentially be competitive with L-shaped on large-scale problems.

The projected subgradient method is a general-purpose method for non-smooth convex optimization but it converges slowly, even with a carefully selected stepsize. It has therefore been difficult to devise subgradient schemes that are competitive with the L-shaped method. On the other hand, the last few decades have witnessed significant advances in the theory and practice of gradient-based methods for minimization of *smooth* convex functions. Several algorithms with optimal (worst-case) complexity guarantees have been developed [13], [14], typically using acceleration techniques inspired by Nesterov's fast gradient scheme [15] and Polyak's heavy ball method [16]. While adding essentially no extra computation or memory, these techniques tend to speed-up the practical convergence of the original method significantly. However, these fast methods are designed to work on smooth problems. The smoothing technique [17] is an attempt to accelerate the convergence also for non-differentiable problems. The basic idea is to approximate the original problem by a smooth one, which is then solved by a (fast) gradient method. With the right amount of smoothing, this approach enjoys strong theoretical convergence guarantees and often displays much better practical performance than the projected subgradient method. This makes smoothing a promising approach for the design of more scalable stochastic programming solvers.

In this work, we develop a fast smoothing scheme for two-stage stochastic programs based on the Moreau envelope [18]. Compared to similar approaches [17], [19], the smoothing is applied to the primal form of the scenario subproblems, which simplifies the implementation significantly. We derive problem-dependent approximation bounds and a strong worst-case complexity guarantee when the smooth approximation is paired with a fast gradient method. The theoretical results highlight a trade-off between convergence speed and accuracy, affected by the degree of smoothing. To accelerate the computation of high-accuracy solutions,

Martin Biel, Vien V. Mai, and Mikael Johansson are with the Division of Decision and Control Systems, School of EECS, KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden. Emails: {`mbiel`, `maivv`, `mikaelj`}`@kth.se`.

we suggest a hybrid approach that initially uses the fast smoothing approach and then switches to a (warm-started) L-shaped which then rapidly finds the full accuracy solution. We evaluate an efficient implementation of this method in our software framework for stochastic programming [20]. The numerical experiments are performed in a multi-node setup with stochastic programs distributed over 32 worker nodes. The smoothing algorithm reduces the execution time of L-shaped by a factor 2-10 on a range of applied problems. Moreover, the hybrid approach achieves essentially the same performance without sacrificing accuracy.

II. Preliminaries

For the remainder of the paper, we consider finite two-stage stochastic programs of the form

$$\begin{aligned} \underset{x\in\mathbb{R}^n, y_s\in\mathbb{R}^m}{\text{minimize}} \quad & c^T x + \sum_{s=1}^{N} \pi_s q_s^T y_s \\ \text{subject to} \quad & Ax = b \\ & T_s x + W y_s = h_s, \qquad s = 1,\dots,N \\ & x \geq 0,\ y_s \geq 0, \qquad s = 1,\dots,N, \end{aligned} \tag{2}$$

where $A \in \mathbb{R}^{p\times n}$, $T_s \in \mathbb{R}^{q\times n}$, $s = 1,\dots,N$ and $W \in \mathbb{R}^{q\times m}$. Scenario-dependent data $\xi_s = \begin{pmatrix} q_s & T_s & h_s \end{pmatrix}^T$ is drawn with probability π_s from a discrete sample space Ω. For problems with infinite sample space, the finite form (2) above can be used to approximate (1) using sample-based techniques (see [21], [22] for details).

By introducing $\mathcal{X} = \{x \in \mathbb{R}^n \mid Ax = b, x \geq 0\}$, and,

$$\begin{aligned} Q_s(x) = \min_{y_s\in\mathrm{R}^m} \quad & q_s^T y_s \\ \text{s.t.} \quad & W y_s = h_s - T_s x \\ & y_s \geq 0. \end{aligned} \tag{3}$$

we can recast the stochastic program on the following form:

$$\underset{x\in\mathcal{X}}{\text{minimize}} \quad f(x), \tag{4}$$

where

$$f(x) = c^T x + \sum_{s=1}^{n} \pi_s Q_s(x). \tag{5}$$

We restrict our attention to problems with *complete recourse*, i.e. problems in which the positive linear span of W, $\operatorname{pos} W$, is the full space. It then follows that all second-stage subproblems (3) are feasible for every $x \in \mathcal{X}$. Furthermore, the following results hold:

Proposition 2.1 ([1]): Assume that $\operatorname{pos} W = \mathbb{R}^m$. The function $Q(x) = \sum_{s=1}^n \pi_s Q_s(x)$ then satisfies:

(i) $Q(x) < \infty, \quad \forall x \in \mathcal{X}$

(ii) $Q(x)$ is convex in $x, \quad \forall x \in \mathcal{X}$.

(iii) $Q(x)$ is piecewise linear in $x, \quad \forall x \in \mathcal{X}$

These properties are the foundation for the design of efficient algorithms for solving (4).

A. The L-shaped Algorithm

The L-shaped algorithm, originally proposed in [6], decomposes (1) into a master problem and N subproblems (one for each second-stage scenario). Here, we introduce the multi-cut extension suggested in [7], which has better convergence properties. The master problem is given by

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & c^T x + \sum_{s=1}^{N} \theta_s \\ \text{subject to} \quad & Ax = b \\ & \partial Q_{s,k} x + \theta_s \geq q_{s,k}, \qquad s = 1,\dots,N \quad \forall k \\ & x \geq 0, \end{aligned}$$

where the $\partial Q_{s,k} x + \theta_s \geq q_{s,k}$ are cutting-plane constraints obtained from solving subproblems of the form (3). It follows from LP duality that $\lambda_{s,k}^T(h_s - T_s x)$, where $\lambda_{s,k}$ is the dual optimizer of (3) at x_k, is a support of $Q_s(x)$ at x_k. Consequently, N cutting planes can be determined each iteration by introducing $\partial Q_{s,k} = \lambda_{s,k}^T T_s$ and $q_{s,k} = \lambda_{s,k}^T h_s$. As the L-shaped algorithm progresses, the master problem is re-solved to generate iterates $x_k, \theta_{s,k}$. The subproblems (3) are then re-solved at the new iterate to generate tighter cutting planes. This is repeated until the gap between the upper bound $Q(x_k) = \sum_{s=1}^n \pi_s Q_s(x_k)$ and lower bound $\theta_k = \sum_{s=1}^n \theta_{s,k}$ reaches a desired relative tolerance. Because W has a finite number of bases, the L-shaped algorithm is finitely convergent [6].

Two well-known drawbacks of the L-shaped algorithm are that initial iterations are typically inefficient and that the final iterations are slowed by the accumulation of ineffective cuts from the early iterations [1]. These issues can be mitigated to a certain extent by regularizing the master problem [8]–[10]. In brief, regularization constrains the candidate search to a neighborhood of the best iterate found so far. This stabilizes the procedure and also enables the algorithm to warm-start from a supplied starting point.

B. Projected Subgradient Descent

The reformulation (4) and Proposition 2.1 reveal that stochastic programs on the form (2) are convex optimization problems. The function $f(x)$ defined in (5) is convex and non-smooth (specifically, piecewise linear) and $\mathcal{X}$ is a closed and convex set. To solve problem (4), the classical projected subgradient method starts from $x_0 \in \mathcal{X}$ and generates a sequence of iterates $\{x_k\}$ defined by:

$$x_{k+1} = \Pi_{\mathcal{X}}(x_k - \gamma_k g_k), \tag{6}$$

where γ_k is the stepsize, g_k is a subgradient of f at x_k and

$$\Pi_{\mathcal{X}}(x) = \arg\min_{z\in\mathcal{X}} \left\{ \|z - x\|_2^2 \right\},$$

is the orthogonal projection of x onto $\mathcal{X}$. When f describes a stochastic program on the form (5), g_k is simply

$$g_k = c - \sum_{s=1}^{m} \pi_s \partial Q_{s,k},$$

where $\partial Q_{s,k}$ is computed as in the L-shaped method, while $\Pi_{\mathcal{X}}(x)$ can be computed by solving a quadratic program. Owing to its generality, the subgradient method suffers from a slow rate of convergence. In particular, procedure (6) needs $O(1/\epsilon^2)$ iterations to find an ϵ-optimal solution when minimizing a Lipschitz continuous and convex f. This is in stark contrast to the optimal complexity $O(1/\sqrt{\epsilon})$ for minimization of smooth convex functions, which is achieved by several variants of Nesterov's acceleration scheme [15].

III. Smoothing procedure

Practical optimization problems are often equipped with favorable structures that can be exploited in the design and analysis of algorithms. Unlike the subgradient method, which treats the objective function as a black box, the smoothing procedure proposed in [17] carefully exploits the problem structure to develop more efficient minimization algorithms. The main idea is to: (i) replace the original nonsmooth function with a nearby smooth one; and (ii) employ an efficient gradient-based method to solve the smooth (but approximate) problem. With this approach, one can find a solution to the original problem in only $O(1/\epsilon)$ iterations compared to $O(1/\epsilon^2)$ of the subgradient method.

A. Smoothing

We now detail our smoothing strategy for the stochastic program (4). For a given scenario s, consider the smooth approximation given by:

$$Q_{s\mu}(x) = \min_{z\in\mathbb{R}^n}\left\{Q_s(z) + \frac{1}{2\mu}\|z-x\|_2^2\right\}, \qquad (7)$$

where $\mu > 0$ is some smoothing parameter. The function can be recognized as the Moreau envelope of $Q_s(x)$. The following results hold.

Proposition 3.1 ([18]): Given a convex function $Q_s(x)$, the smooth approximation $Q_{s\mu}(x)$ satisifes:

(i) $Q_{s\mu}(x)$ has a unique minimizer given by:

$$\text{prox}_{\mu Q_s}(x) = \arg\min_{z\in\mathbb{R}^n}\left\{Q_s(z) + 1/(2\mu)\|z-x\|_2^2\right\}.$$

(ii) $Q_{s\mu}(x)$ is continuously differentiable with gradient

$$\nabla Q_{s\mu}(x) = \frac{1}{\mu}(x - \text{prox}_{\mu Q_s}(x)).$$

(iii) $Q_{s\mu}(x)$ is $(1/\mu)$-smooth:

$$\|\nabla Q_{s\mu}(x) - \nabla Q_{s\mu}(y)\|_2 \le \frac{1}{\mu}\|x-y\|_2 \quad \forall x, y$$

Because $Q_s(x)$ is the optimal value of a linear program, $Q_{s\mu}(x)$ and $\text{prox}_{\mu Q_s}(x)$ can be obtained by solving a quadratic program using general-purpose solvers. Next, we derive approximation bounds for the smoothing function.

Theorem 3.2: There exists a constant $\Gamma > 0$ depending on W such that

$$Q_{s\mu}(x) \le Q_s(x) \le Q_{s\mu}(x) + \mu\Gamma\|q_s\|_2\|T_s\|_2^2.$$

Proof: By the definition of $Q_{s\mu}(x)$, it holds that

$$\begin{aligned} Q_{s\mu}(x) &= \min_{z\in\mathbb{R}^n}\left\{Q_s(z) + 1/(2\mu)\|z-x\|_2^2\right\} \\ &\le Q_s(x) + 1/(2\mu)\|x-x\|_2^2 = Q_s(x), \end{aligned}$$

proving the first inequality. Recall that by strong duality, for any x there exists an optimal dual solution $\hat{\lambda}_s(x)$ such that $Q_s(x) = \hat{\lambda}_s(x)^T(h_s - T_s x)$. Moreover, because Q_s is convex it holds that $Q_s(z) \ge \hat{\lambda}_s(x)^T(h_s - T_s z)$ for all z. Combining these result yields $Q_s(z) - Q_s(x) \ge -\hat{\lambda}_s(x)^T(z-x)$. Now,

$$\begin{aligned} Q_{s\mu}(x) - Q_s(x) &= \min_{z\in\mathbb{R}^n}\left\{Q_s(z) - Q_s(x) + \frac{1}{2\mu}\|z-x\|_2^2\right\} \\ &\ge \min_{z\in\mathbb{R}^n}\left\{-\hat{\lambda}_s(x)^T T_s(z-x) + \frac{1}{2\mu}\|z-x\|_2^2\right\} \\ &= -\frac{\mu}{2}\left\|\hat{\lambda}(x)^T T_s\right\|_2^2 \\ \Leftrightarrow Q_s(x) &\le Q_{s\mu}(x) + \frac{\mu}{2}\|T_s\|_2^2\left\|\hat{\lambda}_s(x)\right\|_2^2 \end{aligned}$$

where the final equality is obtained by recognizing that $z - x = \mu\hat{\lambda}_s(x)^T T_s$ is the optimizer of the quadratic minimization problem. To remove the dependence on the specific dual solution $\hat{\lambda}_s(x)$, we consider

$$\begin{aligned} \underset{\lambda_s\in\mathbb{R}^q}{\text{maximize}} \quad & \|\lambda_s\|_2^2 \\ \text{subject to} \quad & W^T\lambda_s \le q_s \end{aligned}$$

By invoking Hoffman's Lemma [23], we can show that there exists a constant $\Gamma > 0$ depending on W such that

$$\max_{W^T\lambda_s\le q_s}\|\lambda_s\|_2^2 \le 2\Gamma\|q_s\|_2.$$

See [19] for a derivation of this intermediate step. Combining the above results yields

$$Q_s(x) \le Q_{s\mu}(x) + \mu\Gamma\|q_s\|_2\|T_s\|_2^2,$$

proving the second inequality. ■

Applying the smooth approximation to all scenarios yields the following smooth objective function:

$$f_\mu(x) = c^T x + \sum_{s=1}^n \pi_s Q_{s\mu}(x),$$

which by Theorem 3.2 satisfies

$$f_\mu(x) \le f(x) \le f_\mu(x) + \mu\Gamma\bar{q}\bar{T}, \qquad (8)$$

where

$$\bar{q} = \sum_{s=1}^n \pi_s\|q_s\|_2, \quad \bar{T} = \sum_{s=1}^n \pi_s\|T_s\|_2^2.$$

It is also easy to verify that the gradient $\nabla f_\mu(x)$, given by

$$\nabla f_\mu(x) = c + \sum_{s=1}^n \pi_s\nabla Q_{s\mu}(x),$$

is Lipschitz continuous with constant $1/\mu$. Thus, the value μ controls the trade off between the quality of the approximation in (8) and the smoothness of $\nabla f_\mu(x)$.

Having established the smooth approximation f_μ for f, it remains to solve the following convex problem:

$$\underset{x\in\mathcal{X}}{\text{minimize}} \quad f_\mu(x). \qquad (9)$$

The smoothing framework does not specify how to choose a particular method for solving (9) as actual performance is often problem-dependent. Nevertheless, to obtain optimal theoretical complexity bounds, smoothing is often coupled with an accelerated gradient method. Thus, we assume access to an efficient first-order method $\mathcal{M}$ for solving the optimization problem:

$$\underset{x\in\mathcal{X}}{\text{minimize}} \quad \varphi(x),$$

where φ is a convex and L-smooth function, and $\mathcal{X}$ is a closed and convex set. Specifically, there should exist a constant Λ such that the method generates a sequence $\{x_k\} \in \mathcal{X}$ satisfying

$$\varphi(x_k) - \varphi^\star \leq L\Lambda/k^2,$$

where $\varphi^\star = \min_{x\in\mathcal{X}} \varphi(x)$. For example, if we choose $\mathcal{M}$ to be the FISTA method [24] with the constant stepsize $1/L = \mu$, the resulting algorithm can be summarized as:

$$\begin{aligned}
x_{k+1} &= \Pi_{\mathcal{X}}\Big(-\mu c + \sum_{s=1}^{n} \pi_s \operatorname{prox}_{\mu Q_s}(y_k)\Big)\\
t_{k+1} &= \Big(1 + \sqrt{1 + 4t_k^2}\Big)/2\\
y_{k+1} &= x_{k+1} + \frac{t_k - 1}{t_{k+1}}\left(x_{k+1} - x_k\right),
\end{aligned}$$

where $t_0 = 1$, $x_0 = y_0 \in \mathcal{X}$. We can now establish the following convergence result:

Theorem 3.3: Let $\{x_k\}_{k=1}^{\infty}$ be generated by applying the efficient method $\mathcal{M}$ on the smooth approximation problem

$$\underset{x\in\mathcal{X}}{\text{minimize}} \quad f_\mu(x).$$

Then

$$f(x_k) - f(x^\star) \leq \mu\Gamma\bar{q}\bar{T} + \frac{\Lambda}{\mu k^2}.$$

In particular, if $\mu = \varepsilon/(2\Gamma\bar{q}\bar{T})$, then $f(x_k) - f^\star \leq \varepsilon$ after

$$k = \frac{2\sqrt{\Gamma\bar{q}\bar{T}\Lambda}}{\varepsilon}$$

iterations.

Remark 3.1: The theorem implies that the suboptimality gap converges to a residual error (proportional to μ) when k is sufficiently large. Moreover, with a proper choice of μ, the iteration complexity is of the order $O(1/\epsilon)$, which is better than the well-known $O(1/\epsilon^2)$ of the subgradient descent method. The choice of μ will affect both the convergence speed and the residual error.

Proof: Since f_μ is $(1/\mu)$-smooth, executing k iterations of $\mathcal{M}$ yields

$$f_\mu(x_k) - f_\mu^\star \leq \Lambda/(\mu k^2), \tag{10}$$

where $f_\mu^\star = \min_{x\in\mathcal{X}} f_\mu(x)$. Now, let $x^\star$ be an optimal solutions of (4), we have

$$\begin{aligned}
f(x_k) - f(x^\star) = f(x_k) - f_\mu(x_k) + f_\mu(x_k) - f_\mu^\star\\
+ f_\mu^\star - f(x^\star).
\end{aligned} \tag{11}$$

Note that $f_\mu^\star \leq f_\mu(x^\star)$ and by Theorem 3.2:

$$f_\mu(x^\star) \leq f(x^\star) \quad \text{and} \quad f(x_k) - f_\mu(x_k) \leq \mu\Gamma\bar{q}\bar{T}. \tag{12}$$

Combining (10)–(12) thus yields

$$f(x_k) - f(x^\star) \leq \mu\Gamma\bar{q}\bar{T} + \Lambda/(\mu k^2). \tag{13}$$

Finally, plugging $\mu = \epsilon/(2\Gamma\bar{q}\bar{T})$ and $k \geq 2\sqrt{\Gamma\bar{q}\bar{T}\Lambda}/\epsilon$ into (13) completes the proof. ■

B. A hybrid approach

The result in Theorem 3.3 suggests a trade-off between speed and accuracy in the choice of the smoothing parameter μ. In the interest of devising a practical and efficient method, we propose a hybrid approach. Regularizing the L-shaped method, using for example any of the bundle procedures suggested in [8]–[10], allows warm-starting the method from a supplied starting point x_0. We propose using larger values of μ and then run the smoothing procedure until it stalls with some error. At this point, we apply a regularized L-shaped procedure from the current iterate. This approach addresses the drawbacks of both the smoothing procedure and the L-shaped algorithm. It could allow more efficient choices of the smoothing parameter μ as the consequent objective error can be mitigated by L-shaped. In turn, we address the fact that initial iterations of the L-shaped algorithm are often inefficient by providing a much stronger starting point from the smoothing procedure. Also, we do not have to handle ineffective cuts from early iterations of the L-shaped procedure, which improves scalability. These improvements are demonstrated practically in our numerical experiments shown in Section V, where our hybrid approach consistently outperforms L-shaped without sacrificing accuracy.

IV. Implementation

We implement the smoothing procedure in our general-purpose framework for stochastic programming, `StochasticPrograms.jl` [20], written in the Julia programming language. It provides a domain-specific language for stochastic programming, as exemplified in Listing 1, leveraged by the algebraic modeling language JuMP [25]. Moreover, the framework has distributed capabilities. Instantiated stochastic programs can be distributed over multiple worker nodes and then be solved in parallel using specialized solvers.

Listing 1

Example declaration of a stochastic program.

```
@stochastic_model begin
    @stage 1 begin
        @decision(model, x[i in 1:10] >= 0)
        @objective(model, Min, sum(x))
    end
    @stage 2 begin
        @parameters W
        @uncertain q T h
        @recourse(model, y[j in 1:5] >= 0)
        @objective(model, Max, q⋅y)
        @constraint(model, T * x  + W * y .== h)
    end
end
```

The value of any first-stage variable annotated with `@decision` is made available in each second-stage instance through fixed variables. This allows us to efficiently formulate (3). Consequently, we can calculate $Q_s(x)$ and the remaining quantities required to implement the L-shaped algorithm. We can also seamlessly unfix the first-stage variables and add penalty terms to acquire the form (7) required for the smoothing procedure. This is the strength in implementing smoothing using the primal form. It can effortlessly be added to our existing framework, which allows domain experts to formulate stochastic programs with expressive syntax.

V. Numerical Experiments

In this section, we outline our experimental setup and present the results from our large-scale experiments.

A. Experimental Setup

We evaluate the smoothing procedure on a collection of applied problems presented in [22]. The problems are openly available[1] in the SMPS format, which is supported by the software framework. Reasonably tight confidence intervals are obtained for all test problems using a sample size of 5000 [22]. We, therefore, construct sampled instances of each problem with 5000 scenarios when evaluating algorithm performance. We provide a summary of the test set problems and their respective problem dimensions in Table I.

Name	**Application**	**# Variables**	**# Constraints**
`gbd`	Aircraft allocation	25 004	50 017
`20term`	Vehicle assignment	620 003	3 820 064
`ssn`	Telecom network design	875 001	3 530 089

TABLE I

Test set description displaying the name and application of each problem as well as dimensions of the (2) problem instance.

The experiments are performed in a multi-node setup. The master node is a laptop computer with a 2.6 GHz Intel Core i7 processor and 16 GB of RAM. We spawn workers on a remote multi-core machine with two 3.1 GHz Intel Xeon processors (total 32 cores) and 128 GB of RAM. Throughout, the Gurobi optimizer [26] is used to solve subproblems.

For each sampled test problem we carry out the following procedure. First, we apply the L-shaped algorithm and measure the time taken to converge to optimality within a relative tolerance of 1×10^{-2}. Level-set decomposition [10] is used to accelerate the procedure. We then benchmark the smoothing procedure, where FISTA is used to solve (9), on the same problem. Suitable values for the smoothing parameter μ and the stepsize γ are determined by a crude grid-search. Notably, we find that tuning the stepsize is beneficial for performance even when running the FISTA scheme. Finally, we benchmark the hybrid approach. That is, we run the smoothing procedure until progress slows down upon which we warm-start a regularized L-shaped procedure from the current iterate. Each experiment is repeated five times to reduce background noise from operating system task switching. We consider the median time when reporting the results as it is less sensitive to outliers. All experiments are run from the same randomized starting point. In both algorithms, the subproblems are solved in parallel on the 32 worker nodes. The resulting optimality cuts or gradients are communicated back to the master node which performs the respective iterate update. For a thorough introduction to distributed L-shaped methods, we refer to our earlier work [11].

[1] `http://pages.cs.wisc.edu/~swright/stochastic/sampling/`

B. Results

The numerical results are summarized in Table II. In addition, we visualize the solution progress for each problem in Fig. 1, Fig. 2, and Fig. 3. Overall, the gradient schemes outperform L-shaped in all three problem instances, reducing the running time by a factor of 2-10. The smoothing procedure makes fast initial progress but does not reach the same solution accuracy as the L-shaped method. Notably, we used a larger value 1×10^{-2} of the smoothing parameter μ on the `ssn` problem, because it gave a faster performance. The resulting objective error is higher than the other problems, as expected by the result in Theorem 3.3. Finally, we observe that the hybrid scheme is successful in reaching the same relative tolerance at L-shaped while retaining a significant speedup from the smoothing procedure.

Problem	**L-shaped**	**Smoothing**		**Hybrid**	
	T [s]	T [s]	ϵ [%]	T [s]	ϵ [%]
`gbd`	20.77	4.93	1.71	8.73	1.1×10^{-4}
`20term`	296.38	22.34	1.4	46.39	3.5×10^{-5}
`ssn`	106.87	48.21	13.72	60.07	1.7×10^{-2}

TABLE II

Experimental result summary. Displays the time T required for L-shaped to reach a relative tolerance of 1×10^{-2}. Also shows the time T required by the smoothing scheme and the hybrid scheme to reach the resulting objective tolerance ϵ relative to the L-shaped value.

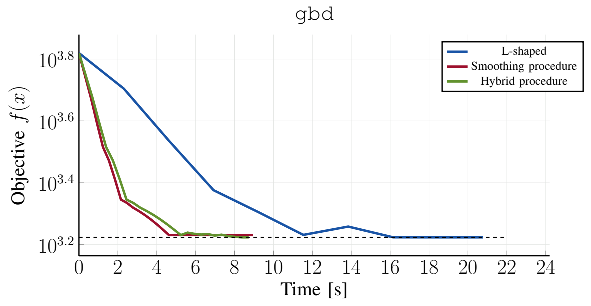

Fig. 1. Algorithm performance comparison on the `gbd` problem. The smoothing procedure is configured by $\mu = 1 \times 10^{-4}$ and $\gamma = 0.01$.

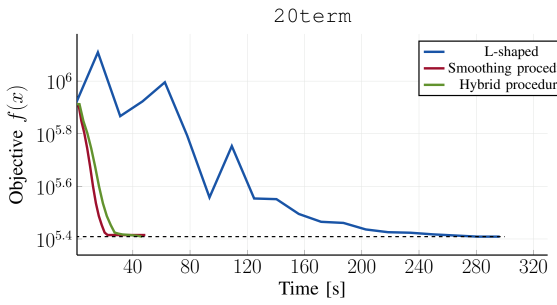

Fig. 2. Algorithm performance comparison on the `20term` problem. The smoothing procedure is configured by $\mu = 1 \times 10^{-4}$ and $\gamma = 0.01$.

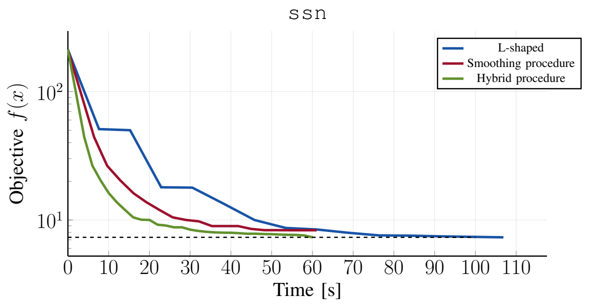

Fig. 3. Algorithm performance comparison on the `ssn` problem. The smoothing procedure is configured by $\mu = 1 \times 10^{-2}$ and $\gamma = 75$.

VI. Concluding remarks

In summary, we have presented a fast smoothing scheme for solving two-stage stochastic programs. The primal formulation through the Moreau envelope allowed us to effortlessly implement the method as an extension in our software framework `StochasticPrograms.jl`. We have also given a problem-dependent approximation bound in Theorem 3.2 and a complexity result when the smooth approximation is combined with a fast gradient method like FISTA in Theorem 3.3. We showcase strong results in distributed numerical experiments. The smoothing procedure consistently outperforms a regularized L-shaped algorithm. The hybrid scheme achieves the same accuracy as L-shaped but still offers a considerable reduction in solution speed. This indicates a potential resurgence of gradient-based methods for solving stochastic programs.

We aim to further explore accelerated gradient methods for efficiently solving stochastic programs. Also, the promising results of our hybrid approach suggest another direction of future work. A more advanced implementation could entail switching between L-shaped iterations and gradient iterations. This could potentially yield further improvements than the current version that only switches once from the smoothing scheme to L-shaped.

References

[1] J. R. Birge and F. Louveaux, *Introduction to Stochastic Programming*. Springer New York, 2011.

[2] R. Louca and E. Bitar, "Stochastic ac optimal power flow with affine recourse," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 2431–2436.

[3] H. Shi, Y. Chu, and F. You, "Integrated planning, scheduling, and dynamic optimization for continuous processes," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 388–393.

[4] R. Shone, K. Glazebrook, and K. G. Zografos, "Applications of stochastic modeling in air traffic management: Methods, challenges and opportunities for solving air traffic problems under uncertainty," *European J. of Operational Research*, vol. 292, no. 1, pp. 1–26, 2021.

[5] C. G. Petra, O. Schenk, and M. Anitescu, "Real-Time Stochastic Optimization of Complex Energy Systems on High-Performance Computers," *Computing in Science Engineering*, vol. 16, no. 5, pp. 32–42, 2014.

[6] R. Van Slyke and R. Wets, "L-Shaped Linear Programs with Applications to Optimal Control and Stochastic Programming," *SIAM Journal on Applied Mathematics*, vol. 17, no. 4, pp. 638–663, 1969.

[7] J. R. Birge and F. V. Louveaux, "A multicut algorithm for two-stage stochastic linear programs," *European Journal of Operational Research*, vol. 34, no. 3, pp. 384–392, 1988.

[8] A. Ruszczyński, "A regularized decomposition method for minimizing a sum of polyhedral functions," *Mathematical Programming*, vol. 35, no. 3, pp. 309–333, 1986.

[9] J. Linderoth and S. Wright, "Decomposition Algorithms for Stochastic Programming on a Computational Grid," *Computational Optimization and Applications*, vol. 24, no. 2-3, pp. 207–250, 2003.

[10] C. I. Fábián and Z. Szőke, "Solving two-stage stochastic programming problems with level decomposition," *Computational Management Science*, vol. 4, no. 4, pp. 313–353, 2006.

[11] M. Biel and M. Johansson, "Distributed L-shaped algorithms in Julia," in *2018 IEEE/ACM Parallel Applications Workshop, Alternatives To MPI (PAW-ATM)*. IEEE, 2018.

[12] Y. Ermoliev, *Stochastic quasigradient methods*. Springer-Verlag, Berlin, 1988, pp. 141–186.

[13] Y. Nesterov *et al.*, *Lectures on convex optimization*. Springer, 2018, vol. 137.

[14] A. Beck, *First-order methods in optimization*. SIAM, 2017, vol. 25.

[15] Y. E. Nesterov, "A method for solving the convex programming problem with convergence rate o (1/kˆ 2)," in *Dokl. akad. nauk Sssr*, vol. 269, 1983, pp. 543–547.

[16] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.

[17] Y. Nesterov, "Smooth minimization of non-smooth functions," *Mathematical programming*, vol. 103, no. 1, pp. 127–152, 2005.

[18] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex analysis and minimization algorithms*. Springer science & business media, 1993, vol. 305.

[19] S. Ahmed, "Smooth minimization of two-stage stochastic linear programs," *Manuscript, Georgia Institute of Technology*, 2006.

[20] M. Biel and M. Johansson, "Efficient stochastic programming in Julia," *arXiv preprint arXiv:1909.10451*, 2019, submitted for consideration to Mathematical Programming Computation. Under review.

[21] W.-K. Mak, D. P. Morton, and R. Wood, "Monte Carlo bounding techniques for determining solution quality in stochastic programs," *Operations Research Letters*, vol. 24, no. 1, pp. 47 – 56, 1999.

[22] J. Linderoth, A. Shapiro, and S. Wright, "The empirical behavior of sampling methods for stochastic programming," *Annals of Operations Research*, vol. 142, no. 1, pp. 215–241, 2006. [Online]. Available: https://doi.org/10.1007/s10479-006-6169-8

[23] A. J. Hoffman, "On approximate solutions of systems of linear inequalities," *Journal of Research of the National Bureau of Standards*, vol. 49, pp. 263–265, 1952.

[24] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[25] I. Dunning, J. Huchette, and M. Lubin, "JuMP: A modeling language for mathematical optimization," *SIAM Review*, vol. 59, no. 2, pp. 295–320, 2017.

[26] G. Optimization, "Gurobi optimizer reference manual," 2020, http://www.gurobi.com.