Journal of Computational and Applied Mathematics 290 (2015) 196–208

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Improving the performance of Stochastic Dual Dynamic Programming

Vitor L. de Matos [a,*], Andy B. Philpott [b], Erlon C. Finardi [c]

[a] *Plan4 Engenharia, Rua Maria Luiza Agostinho, n 92, Itacorubi, Florianopolis, Brazil*
[b] *Electric Power Optimization Centre, University of Auckland, New Zealand*
[c] *Laboratório de Planejamento de Sistemas de Energia Elétrica, Universidade Federal de Santa Catarina, Brazil*

HIGHLIGHTS

- Presents two tree traversing strategies for SDDP.
- Presents three cut selection algorithms to improve SDDP performance.
- Considers the whole Brazilian Power System in our computational results.
- Significant reduction in computational time without compromising the policy.

ARTICLE INFO

Article history:
Received 7 January 2013
Received in revised form 8 January 2015

Keywords:
Stochastic programming
Stochastic Dual Dynamic Programming
Cut selection
Hydrothermal scheduling

ABSTRACT

This paper is concerned with tuning the Stochastic Dual Dynamic Programming algorithm to make it more computationally efficient. We report the results of some computational experiments on a large-scale hydrothermal scheduling model developed for Brazil. We find that the best improvements in computation time are obtained from an implementation that increases the number of scenarios in the forward pass with each iteration and selects cuts to be included in the stage problems in each iteration. This gives an order of magnitude decrease in computation time with little change in solution quality.

1. Introduction

The Stochastic Dual Dynamic Programming (SDDP) algorithm of Pereira and Pinto [1] is a technique for attacking multi-stage stochastic linear programs that have a stage-wise independence property that makes them amenable to dynamic programming. This method approximates the future cost function of dynamic programming using a piecewise linear outer approximation, defined by cutting planes computed by solving linear programs. This helps to mitigate the curse of dimensionality that arises from discretizing the state variables. The intractability arising from a branching scenario tree is avoided by essentially assuming stage-wise independent uncertainty. This allows cuts to be shared between different states, effectively collapsing the scenario tree. Although it was developed over twenty years ago, and has been cited over the years in many applied papers, SDDP has received some recent attention in the mathematical programming literature [2–7] that explores the mathematical properties of this method, in some cases extending it to deal with risk-averse objective functions.

This paper is concerned with some of the implementation details of SDDP algorithms. By carrying out some computational tests on a real application, we attempt to draw some conclusions about how the basic method might be improved by tuning

* Corresponding author. Tel.: +55 4899812565.
E-mail addresses: vitor@plan4.com.br (V.L. de Matos), a.philpott@auckland.ac.nz (A.B. Philpott), erlon.finardi@ufsc.br (E.C. Finardi).

http://dx.doi.org/10.1016/j.cam.2015.04.048

it to build a near optimal policy in the shortest computation time. There are basically two techniques for tuning the algorithm that we shall investigate. The first concerns how one should visit the scenarios in the SDDP algorithm. The classical version of SDDP [1] samples a fixed number of scenarios for each "forward pass". We compare this with alternative strategies that traverse one scenario at a time [5], as well as one that increases the number of scenarios per pass as the algorithm proceeds (as discussed in [7]).

In tandem with scenario selection, we investigate several strategies for selecting cuts to include in the linear programming problems that are solved at each stage. This can lead to a dramatic decrease in computation time with little degradation in the performance of the policies obtained. As a consequence, cut selection strategies are crucial for solving real-life hydrothermal scheduling problems using SDDP-type algorithms within a computation time that must be kept modest so that models can be solved frequently by the Independent System Operator and by energy companies to provide support to their decisions.

The paper is laid out as follows. In Section 2 we recall the basic algorithm for SDDP. Section 3 describes the three different tree-traversal strategies we shall test and Section 4 describes methods we use for selecting cuts. Section 5 then shows the results of applying these strategies to a test problem that is derived from a long-term model of the Brazilian electricity system. We make our final remarks in Section 6.

2. Stochastic dual dynamic programming

To describe how SDDP works, we consider a class of stochastic linear programs that have T stages, denoted $t = 1, 2, \ldots, T$, in each of which a random right-hand-side vector $b_t(\omega_t) \in \mathbb{R}^m$ has a finite number of realizations defined by $\omega_t \in \Omega_t$. We assume that the outcomes ω_t are stage-wise independent, and that Ω_1 is a singleton, so the first-stage problem is

$$\begin{aligned} z = \min\ & c_1^\top x_1 + \mathbb{E}[Q_2(x_1, \omega_2)] \\ \text{s.t.}\ & A_1 x_1 = b_1, \\ & x_1 \geq 0, \end{aligned} \tag{1}$$

where $x_1 \in \mathbb{R}^n$ is the first stage decision and $c_1 \in \mathbb{R}^n$ a cost vector, A_1 is a $m \times n$ matrix, and $b_1 \in \mathbb{R}^m$.

We denote by $Q_2(x_1, \omega_2)$ the second stage costs associated with decision x_1 and realization $\omega_2 \in \Omega_2$. The problem to be solved in the second and later stages t, given state x_{t-1} and realization ω_t, can be written as

$$\begin{aligned} Q_t(x_{t-1}, \omega_t) = \min\ & c_t^\top x_t + \mathbb{E}[Q_{t+1}(x_t, \omega_{t+1})] \\ \text{s.t.}\ & A_t x_t = b_t(\omega_t) - E_t x_{t-1}, \quad [\pi_t(\omega_t)] \\ & x_t \geq 0, \end{aligned} \tag{2}$$

where $x_t \in \mathbb{R}^n$ is the decision in stage t, c_t its cost, and A_t and E_t denote $m \times n$ matrices. Here $\pi_t(\omega_t)$ denotes the dual variables of the constraints. In stochastic control terminology $\mathbb{E}[Q_{t+1}(x_t, \omega_{t+1})]$ represents a Bellman function. In the last stage we assume either that $\mathbb{E}[Q_{T+1}(x_T, \omega_{T+1})] = 0$, or that there is a convex polyhedral function that defines the expected future cost after stage T.

The information structure of the problem defined by (1) and (2) is illustrated in Fig. 1 in the form of a scenario tree. A scenario tree starts from a root node that defines the first stage decisions (1), and as we move forward in time it branches in the set of possible realizations for the random variable. As one can notice from the figure, the number of variables and constraints in the multistage stochastic programming problem grows very fast with the number of children nodes (branches) and stages, which makes it impossible to solve in most real applications. In order to overcome this difficulty, the SDDP algorithm relies on a stage-wise independence assumption to enable a dynamic programming simplification. By sampling scenarios, a policy for this dynamic program is computed that is close to optimal for the values of the state variables that are visited by this policy.

Before defining the sampling procedure, it is important to understand that the algorithm aims at building a policy that is defined at stage t by a polyhedral outer approximation of $\mathbb{E}[Q_{t+1}(x_t, \omega_{t+1})]$ resulting in an approximate value function $\mathcal{Q}_t(x_{t-1}, \omega_t)$. The outer approximation is constructed using cutting planes called Benders cuts, or just *cuts*. In other words in each tth-stage problem, $\mathbb{E}[Q_{t+1}(x_t, \omega_{t+1})]$ is replaced by the variable θ_{t+1} which is constrained by the set of linear inequalities

$$\theta_{t+1} + \bar{\pi}_{t+1,k}^\top E_{t+1} x_t \geq \bar{g}_{t+1,k} \quad \text{for } k = 1, 2, \ldots, K, \tag{3}$$

where K is the number of cuts. Here $\bar{\pi}_{t+1,k} = \mathbb{E}[\pi_{t+1}(\omega_{t+1})]$, which defines the gradient $-\bar{\pi}_{t+1,k}^\top E_{t+1}$ and the intercept $\bar{g}_{t+1,k}$ for cut k in stage t, where

$$\bar{g}_{t+1,k} = \mathbb{E}[\mathcal{Q}_{t+1}(x_t^k, \omega_{t+1})] + \bar{\pi}_{t+1,k}^\top E_{t+1} x_t^k.$$