Solution of Emission Management Problem

Václav Kozmík, Martin Šmíd[1]

Abstract

Optimal covering of emissions stemming from random production is a multistage stochastic programming problem. Solving it in a usual way - by means of deterministic equivalent – is possible only given an unrealistic approximation of random parameters. There exists an efficient way of solving multistage problems – stochastic dual dynamic programming (SDDP); however, it requires the inter-stage independence of random parameters, which is not the case which our problem. In the paper, we discuss a modified version of SDDP, allowing for some form of interstage dependence.

Keywords

Multistage stochastic programming, Emission management, SDDP, time dependence.

JEL Classification: C44

1. Introduction

In order to reduce CO_2 emissions, EU, as well as some other countries, introduced the Cap-and-Trade system: Each year, companies are obliged to hand out emission allowances, one for each ton of CO_2 they emitted in the previous year. The allowances may be bought from the government in auctions, the companies may trade with them on various secondary markets; moreover, from protectionist reasons, governments grant some allowances to the companies for free. The allowances may be banked (saved for later). In addition to spots (the allowances themselves), various derivatives are traded on secondary markets, including futures and options. Thus, optimal covering of the emissions, amount of which is usually uncertain, is a complex problem.

[1]Institute of Information Theory and Automation, The Czech Academy of Sciences, Pod Vodárenskou věží 4, 182 08 Praha 8. Czech Republic. e-mail: smid@utia.cas.cz.
This work was supported by grant No. GA 16-01298S of the Czech Science Foundation.

2. Solution of the Optimal Emission Covering Problem

As the problem of optimal emission covering is dynamic with uncertain parameters, its mathematical description naturally leads to multi-stage stochastic programming problems, random parameters of which include at least the emitted amounts of CO_2 and the prices of the allowances spots and derivatives (see [9, 10]). Unfortunately, solution of such a problem by standard techniques is virtually impossible. Solving the problem by means of deterministic equivalent requires unrealistic approximations, otherwise it would lead to extremely huge equivalent problems. Dynamic programming, on the other hand, cannot be used due to an extensively rich state space.

Stochastic dual decomposition (SDDP) – a well known efficient technique for solving multi-stage problems – cannot be used, too, as it requires the stage-wise independence of the underlying random processes. This assumption, however, is not met by the emission prices, time series of which are are non-stationary. Although the non-stationarity can be circumvented by working with returns, see [3], such a trick is impossible within the emission management problem because there is anoother random parameter - the emitted amounts – involved.

There is, however, a way of reconciling the SDDP with dependence: using a hidden Markov model. In particular, we can assume the price process, to be a sum of a (non-homogeneous) Markov chain (e.g. one with a single initial state $\{0\}$, states $\{-1, 1\}$ at time one, $\{-2, 0, 2\}$ at time two, etc.) and an i.i.d. variable. Given this setting, we can use the modification of the SDDP for Markov Chains, described below, to solve the problem. Even though the (unconditional) distribution of the (log-)random parameter is not then a random walk as it would be if we modelled the price process standard way, it is similar to it (and may be made arbitrarily close by making the MC denser; this, however, would make the solution slower).

3. Stochastic dual dynamic programming with Markov Chains

Stochastic dual dynamic programming algorithm is very popular for solving multi-stage stochastic programs. On of the most common risk-averse formulations, nested CVaR model, presented for instance in [3] or [8] can we written in terms of dynamic programming equations as follows:

$$\begin{aligned} \min_{\boldsymbol{x}_1, u_1} \; & \boldsymbol{c}_1^\top \boldsymbol{x}_1 + \lambda_2 u_1 + \mathcal{Q}_2(\boldsymbol{x}_1, u_1) \\ \text{s.t. } & \boldsymbol{A}_1 \boldsymbol{x}_1 = \boldsymbol{b}_1 \\ & \boldsymbol{x}_1 \geq 0 \end{aligned} \tag{1}$$

with the recourse value $Q_t(\boldsymbol{x}_{t-1}, \boldsymbol{\xi}_t)$ at stage $t = 2, \ldots, T$ given by:

$$\begin{aligned} Q_t(\boldsymbol{x}_{t-1}, \boldsymbol{\xi}_t) = \min_{\boldsymbol{x}_t, u_t} \; & \boldsymbol{c}_t^\top \boldsymbol{x}_t + \lambda_{t+1} u_t + \mathcal{Q}_{t+1}(\boldsymbol{x}_t, u_t) \\ \text{s.t. } & \boldsymbol{A}_t \boldsymbol{x}_t = \boldsymbol{b}_t - \boldsymbol{B}_t \boldsymbol{x}_{t-1} \\ & \boldsymbol{x}_t \geq 0, \end{aligned} \tag{2}$$

where

$$\mathcal{Q}_{t+1}(\boldsymbol{x}_t, u_t) = \mathbb{E}\left[(1 - \lambda_{t+1}) \, Q_{t+1}(\boldsymbol{x}_t, \boldsymbol{\xi}_{t+1}) + \frac{\lambda_{t+1}}{\alpha_{t+1}} \left[Q_{t+1}(\boldsymbol{x}_t, \boldsymbol{\xi}_{t+1}) - u_t \right]_+ \right]. \tag{3}$$

We use stochastic dual dynamic programming to solve, or rather approximately solve, equations above. SDDP does not operate directly on these equation. Instead, we first form a sample average approximation (SAA) of the model, and SDDP approximately solves that SAA. Thus in our context SDDP forms estimators by sampling within an empirical scenario tree. In the remainder of this article we restrict attention to solving that SAA via SDDP. See Shapiro [7] for a discussion of asymptotics of SAA for multi-stage problems, Philpott and Guan [6] for convergence properties of SDDP, and Chiralaksanakul and Morton [2] for procedures to assess the quality of an SDDP-based policy.

In most common applications of stochastic dual dynamic programming , we assume stage-wise independence of the underlying random process $\boldsymbol{\xi}_t,\ t = 2, \ldots, T$. Following the idea of Philpott and Matos [5] we weaken here the assumption of stage-wise independence and suppose that the underlying random process depends on state of the system, which is a Markov chain. That said, we suppose that in the initial stage, our Markov chain is in the deterministic state denoted $s_1 \in S_1,\ |S_1| = 1$. For second stage, possible states are given by a set S_2 and transition is driven by transition matrix $T_{1,2}$. Generally, transition from stage t to $t+1$ is driven by matrix $T_{t,t+1}$.

We assume that for each stage $t = 2, \ldots, T$ and each state $s_t \in S_T$ there is a known (possibly continuous) distribution P_{t,s_t} of $\boldsymbol{\xi}_t$ and that we have a procedure to sample i.i.d. observations from this distribution. Using this procedure we obtain empirical distributions $\hat{P}_{t,s_t},\ t = 2, \ldots, T,\ s_t \in S_T$. The scenarios generated by this procedure all have the same probabilities, but this is not required by the SDDP algorithm, which also applies to the case where the scenario probabilities differ.

We let $\hat{\Omega}_{t,s_t}$ denote the stage t sample space for state s_t, where $|\hat{\Omega}_{t,s_t}| = D_{t,s_t}$. All possible realizations in stage t are denoted $\hat{\Omega}_t$, $\hat{\Omega}_t = \bigcup_{s_t \in S_t} \hat{\Omega}_{t,s_t}$, with $|\hat{\Omega}_t| = D_t$, $D_t = \sum_{s_t \in S_t} D_{t,s_t}$. We use $j_t \in \hat{\Omega}_t$ to denote a stage t sample point, which we call a stage t scenario. For each sample point, we can determine its actual state in Markov chain, $s(j_t) \in S_t$. We define the mapping $a(j_t) : \hat{\Omega}_t \to \hat{\Omega}_{t-1}$, which specifies the unique stage $t-1$ ancestor for the stage t scenario j_t. Similarly, we use $\Delta(j_t, s_{t+1}) : \hat{\Omega}_t \to 2^{\hat{\Omega}_{t+1,s_{t+1}}}$ and $\Delta(j_t) = \bigcup_{s_{t+1} \in S_{t+1}} \Delta(j_t, s_{t+1})$ to denote the set of descendant nodes for j_t, where $|\Delta(j_t, s_{t+1})| = D_{t+1,s_{t+1}}$ and $|\Delta(j_t)| = D_{t+1}$. The empirical scenario tree therefore has stage t realizations denoted $\boldsymbol{\xi}_t^{j_t},\ j_t \in \hat{\Omega}_t$. At the last stage, we have $\boldsymbol{\xi}_T^{j_T},\ j_T \in \hat{\Omega}_T$, and each stage T scenario corresponds to a full path of observations through each stage of the scenario tree. That is, given j_T, we recursively have $j_{t-1} = a(j_t)$ for $t = T, T-1, \ldots, 2$. For this reason and for notational simplicity, when possible, we suppress the stage T subscript and denote $j_T \in \hat{\Omega}_T$ by $j \in \hat{\Omega}$.

We emphasize using the same set of D_{t,s_t} observations at stage t for state s_t to form the descendant nodes of all N_{t-1} scenarios at stage $t-1$. This ensures the resulting empirical scenario tree has required independence properties. The SDDP algorithm does not apply, for example, to a scenario tree in which we instead use a separate, independent set of i.i.d. observations $\boldsymbol{\xi}_{t,s_t}^1, \ldots, \boldsymbol{\xi}_{t,s_t}^{D_{t,s_t}}$ for each of the stage $t-1$ scenarios.

In order to apply SDDP, we split the recourse values in stage t by their corresponding states and arrive in the setup which resembles the classical stage-wise independent formulation. Let's denote the probability of transition from state s_t to s_{t+1} $p_{t,t+1}$, which is an element of the

matrix $T_{t,t+1}$. The equations (2) and (3) now read as follows:

$$\begin{aligned} Q_t(\boldsymbol{x}_{t-1}, \boldsymbol{\xi}_{t,s_t}) = \min_{\boldsymbol{x}_t, u_t} \; & \boldsymbol{c}_t^\top \boldsymbol{x}_t + \lambda_{t+1} u_t + \mathcal{Q}_{t+1}(\boldsymbol{x}_t, u_t) \\ \text{s.t. } & \boldsymbol{A}_t \boldsymbol{x}_t = \boldsymbol{b}_t - \boldsymbol{B}_t \boldsymbol{x}_{t-1} \\ & \boldsymbol{x}_t \geq 0, \end{aligned} \tag{4}$$

where

$$\mathcal{Q}_{t+1}(\boldsymbol{x}_t, u_t) = \sum_{s_{t+1} \in S_{t+1}} p_{t,t+1} \mathcal{Q}_{t+1}(\boldsymbol{x}_t, u_t, s_{t+1}). \tag{5}$$

and

$$\mathcal{Q}_{t+1}(\boldsymbol{x}_t, u_t, s_{t+1}) = \mathbb{E}_{s_{t+1}} \left[(1 - \lambda_{t+1}) \, Q_{t+1}(\boldsymbol{x}_t, \boldsymbol{\xi}_{t+1,s_{t+1}}) + \frac{\lambda_{t+1}}{\alpha_{t+1}} \left[Q_{t+1}(\boldsymbol{x}_t, \boldsymbol{\xi}_{t+1,s_{t+1}}) - u_t \right]_+ \right]. \tag{6}$$

We give a brief description of the SDDP algorithm in order to give sufficient context for presenting our results. For further related details on SDDP, see [3], [4] and [8]. SDDP applies to the dynamic programming equations (1), (1) and (5). During a typical iteration of the SDDP algorithm, cuts have been accumulated at each stage. These represent a piecewise linear outer approximation of the expected future cost functions, $\mathcal{Q}_{t+1}(\boldsymbol{x}_t, u_t, s_{t+1})$, separately for each possible state. On a forward pass we sample a number of linear paths through the tree. As we solve a sequence of master programs (which we specify below) along these forward paths, the cuts that have been accumulated so far are used to form decisions at each stage. Solutions found along a forward path in this way form a policy, which does not anticipate the future. In fact, the solutions can be found at a node on a sample path via the stage t master program, even before we sample the random parameters at stage $t+1$. The sample mean of the costs incurred along all the forward sampled paths through the tree forms an estimator of the expected cost of the current policy, which is determined by the master programs.

In the backward pass of the algorithm, we add cuts to the collection defining the current approximation of the expected future cost function at each stage. We do this by solving subproblems at the descendant nodes of each node in the linear paths from the forward pass, except in the final stage, T. The cuts collected at any node in stage t apply to all the nodes in that stage, and hence we maintain a *single set of cuts* for each stage, however, separately for each future state s_{t+1}. We let $C_t t + 1$ denote the number of cuts accumulated so far in stage t. This reduction is possible because of our Markov property assumption.

The following model (7) is based on equations above and acts as a master program for its stage $t+1$ descendant scenarios and acts as a subproblem for its stage $t-1$ ancestor:

$$\hat{Q}_t = \min_{\boldsymbol{x}_t, u_t, \theta_t} \quad \boldsymbol{c}_t^\top \boldsymbol{x}_t + \lambda_{t+1} u_t + \theta_t \tag{7a}$$

$$\text{s.t.} \quad \boldsymbol{A}_t \boldsymbol{x}_t = \boldsymbol{b}_t - \boldsymbol{B}_t \boldsymbol{x}_{t-1} \quad : \boldsymbol{\pi}_t \tag{7b}$$

$$\theta_t \geq \sum_{s_{t+1} \in S_{t+1}} p_{t,t+1} \theta_{t,s_{t+1}} \tag{7c}$$

$$\theta_{t,s_{t+1}} \geq \hat{\mathcal{Q}}^j_{t+1,s_{t+1}} + \left(\mathfrak{g}^j_{t+1,s_{t+1}} \right)^\top \left[(\boldsymbol{x}_t, \, u_t) - \left(\boldsymbol{x}^j_t, \, u^j_t \right) \right], \quad \forall s_{t+1} \in S_{t+1}, j = 1, \ldots, C_{t,s_{t+1}} \tag{7d}$$

$$\boldsymbol{x}_t \geq 0 \tag{7e}$$

Decision variable θ_t in the objective function (7a), coupled with cut constraints in (7c) and (7d), forms the outer linearisation of the recourse function $\mathcal{Q}_{t+1}(\boldsymbol{x}_t, u_t)$ from model (4) and equations (5) – (6) . The structural and nonnegativity constraints in (7b) and (7e) simply repeat the same constraints from model (4). In the final stage T, we omit the cut constraints and the θ_T term.

As we indicate in constraint (7b), we use $\boldsymbol{\pi}_t$ to denote the dual vector associated with the structural constraints. As detailed in the articles [3] and [8], this dual vector is used to develop the cuts in the backward pass of the SDDP algorithm. For simplicity in stating the SDDP algorithm below, we assume we have known lower bounds L_t on the recourse functions.

Algorithm 1. *Stochastic dual dynamic programming algorithm*

1. *Let iteration $k = 1$ and append lower bounding cuts $\theta_t \geq L_t$, $t = 1, \ldots, T-1$.*

2. *Solve the stage 1 master program ($t = 1$) and obtain $\boldsymbol{x}_1^k, u_1^k, \theta_1^k$.*
   Let $\underline{z}_k = \boldsymbol{c}_1^\top \boldsymbol{x}_1^k + \lambda_2 u_1^k + \theta_1^k$.

3. *Forward pass: sample independent paths from $\hat{\Omega}$ and index them by S^k. Each path is sampled by first sampling the next stage state s_{t+1} by corresponding probability and then by sampling a scenario from P_{t,s_t}*

   For all $j \in S^k$ {
   For $t = 2, \ldots, T$ {
   Form and solve $\mathrm{sub}(j_t)$ *to obtain $\left(\boldsymbol{x}_t^{j_t}\right)^k$ and $\left(u_t^{j_t}\right)^k$;*
   }
   }

   Form the upper bound estimator $\overline{z}_k$ with one of the estimators provided in [3].

4. *If a stopping criterion, given $\overline{z}_k$ and $\underline{z}_k$, is satisfied then stop and output first stage solution $\boldsymbol{x}_1 = \boldsymbol{x}_1^k$ and lower bound $\underline{z} = \underline{z}_k$, otherwise continue to step 5.*

5. *Backward pass:*

   For $t = T-1, \ldots, 1$ {
   For all $j \in S^k$ {
   Determine the state in stage $t+1$ in path j: $s_{t+1} = s(j_{t+1})$
   For all descendant nodes $i_{t+1} \in \Delta(j_t, s_{t+1})$ {
   Form and solve $\mathrm{sub}(i_{t+1})$ *to obtain $\hat{Q}_{t+1}^{i_{t+1}}$ and $\boldsymbol{\pi}_{t+1}^{i_{t+1}}$;*
   Calculate subgradient (see [3], [8]);
   }
   Average optimal values and subgradients (see [3], [8]);
   Append the resulting cut to the collection (7d) for stage t;
   }
   }

6. *Let $k = k + 1$ and goto step 2 with extended sets of cuts.*

See Bayraksan and Morton [1] for stopping rules that can be employed in step 4 and Philpott et al. [5] for an alternative upper bound evaluation procedure.

4. Conclusion

We outlined a way how to incorporate non-stationary random parameters into the (Markov) SDDP method. Clearly, there is much to do further, including a discussion about convergence of such algorithm, its accuracy, and numerical study. However, even without these, our paper may give an instruction for practical situations in which a multi-stage model with a non-stationary random parameter has to be solved.

References

[1] BAYRAKSAN, G. and MORTON, D. P. (2011): *A sequential sampling procedure for stochastic programming*, Operations Research 59, pp. 898–913.

[2] CHIRALAKSANAKUL, A. and MORTON, D. P. (2004): *Assessing policy quality in multi-stage stochastic programming*, The Stochastic Programming E-Print Series.

[3] KOZMÍK, V. and MORTON, D. (2013): *Risk-averse Stochastic Dual Dynamic Programming*, Optimization Online.

[4] PEREIRA, M. V. F. and PINTO, L. M. V. G. (1991): *Multi-stage stochastic optimization applied to energy planning*, Mathematical Programming 52, pp. 359–375.

[5] PHILPOTT, A. B., DE MATOS, V. L. and FINARDI, E. C. (2012): *On solving multi-stage stochastic programs with coherent risk measures*, Optimization Online.

[6] PHILPOTT, A. B. and GUAN, Z. (2008): *On the convergence of sampling-based methods for multi-stage stochastic linear programs*, Operations Research Letters 36, pp. 450–455.

[7] SHAPIRO, A. (2003): *Inference of statistical bounds for multistage stochastic programming problems*, Mathematical Methods of Operations Research 58, pp. 57–68.

[8] SHAPIRO, A. (2011): *Analysis of stochastic dual dynamic programming method*, European Journal of Operational Research 209, pp. 63–72.

[9] ŠMÍD, M and ZAPLETAL, F and HANČLOVÁ, J (2017): *Which Carbon Derivatives Are Applicable in Practise? A Case Study of a European Steel Industry*, Kybernetika 53, pp. 1071-1085.

[10] ZAPLETAL, F and ŠMÍD, M and KOPA, M (2018): *Multi-stage emissions management of a steel company*, to appear.