Operations Research Letters 45 (2017) 34–39

Contents lists available at ScienceDirect

Operations Research Letters

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

BBPH: Using progressive hedging within branch and bound to solve multi-stage stochastic mixed integer programs

Jason Barnett[a], Jean-Paul Watson[b], David L. Woodruff[c,*]

[a] Department of Applied Mathematics, University of California Davis, Davis, CA 95616-8609, USA
[b] Discrete Math and Optimization Department, Sandia National Laboratories, Albuquerque, NM 87185, USA
[c] Graduate School of Management, University of California Davis, Davis, CA 95616-8609, USA

ARTICLE INFO

Article history:
Received 29 September 2016
Received in revised form
20 November 2016
Accepted 21 November 2016
Available online 27 November 2016

Keywords:
Stochastic programming
Progressive hedging
Branch and bound

ABSTRACT

Progressive hedging, though an effective heuristic for solving stochastic mixed integer programs (SMIPs), is not guaranteed to converge in this case. Here, we describe BBPH, a branch and bound algorithm that uses PH at each node in the search tree such that, given sufficient time, it will always converge to a globally optimal solution. In addition to providing a theoretically convergent "wrapper" for PH applied to SMIPs, computational results demonstrate that for some difficult problem instances branch and bound can find improved solutions after exploring only a few nodes.

1. Introduction

Spurred by important potential applications, researchers have recently devoted considerable energy to developing methods for solving stochastic mixed integer programs (SMIPs). Progressive hedging (PH), though an effective heuristic for SMIPs, is not guaranteed to converge in the integer case. Here, we propose BBPH, a branch and bound (B&B) algorithm that uses PH at each node of the search tree. Given sufficient time, BBPH will converge to a globally optimal solution. Almost all PH innovations and applications described in the literature can be embedded in this framework.

Originally proposed by Rockafellar and Wets [12], examples of PH in the literature include [3,4,6,9,10].

Branch and bound for SMIPs is not new, of course. For example, [1] proposes a scenario decomposition algorithm for 0–1 stochastic programs, a specific case of our intended general problem class. The dual decomposition algorithm given in [2] uses B&B for a two stage SMIP. A branch-and-fix algorithm for solving multi-stage, stochastic, mixed 0–1 problems is described in [5].

What is new in this contribution is B&B for PH, which can be applied to a broad class of multi-stage SMIPs. A general-purpose software implementation is available as part of the PySP library in the Pyomo software package (www.pyomo.org). Tests of the algorithm and this implementation are described in Section 4. Before describing those tests, we introduce in the remainder of this section the optimization model formulation and the progressive hedging algorithm for multi-stage SMIPs. Section 2 then describes two small examples illustrating the need for branching in PH, which are provided as formal motivation. Our BBPH branch and bound algorithm is described along with a remark whose proof demonstrates its correctness in Section 3.

1.1. A general SMIP optimization model

Let T be the number of decision stages for a multi-stage stochastic program. We will use $t \in 1, \ldots, T$ to index stages, although decision stages do not always correspond to time. To make abstract statements about stochastic programming, we make use of a random variable, which may be vector valued, ξ^t, associated with each decision stage t. When the stages correspond to time, we think of the value(s), ξ^t, during or at the end of, stage $t-1$ depending on the application. Hence, we generally refer to ξ^t only for stage $2, \ldots, T$. Consequently, the decisions for the stage t are made once the random variables for the stages up to and including t are known.

We use the symbol $\vec{\xi}^t$ to refer to the realized values of all random variables up to and including stage t. We refer to a full realization of the uncertainty, i.e.,

$$\vec{\xi}^{\,T} = \left(\xi^t,\ t = 2, \ldots, T\right)$$

as a *scenario*. Often, problem data is a function of ξ and the problem data corresponding to a particular value of ξ is also referred to as a scenario.

* Corresponding author.
E-mail addresses: jpbarnett@math.ucdavis.edu (J. Barnett), jwatson@sandia.gov (J.-P. Watson), dlwoodruff@ucdavis.edu (D.L. Woodruff).

http://dx.doi.org/10.1016/j.orl.2016.11.006
0167-6377/

In abstract formulations, we use $x^t = (u^t, y^t)$ to, respectively, represent the integer and real parts of the decision vector that corresponds to stage t. We use the notation $\vec{x}^{\,t}$ for $1 \leq t \leq T$ to represent the decisions for all stages up to, and including, stage t. We assume that there are functions of the decision variables for the current stage, parameterized by the decisions and random variable realizations known at the time of the decisions. The functions return the objective function value corresponding to the stage for feasible solutions and a very large number for infeasible solutions. For the first stage we write $f_1(x^1)$ and for subsequent stages $f_t(x^t; \vec{x}^{\,t-1}, \vec{\xi}^{\,t})$. The function f_t takes an argument the partial vector corresponding to stage t and as parameters the solution for previous stages and the realization of the random variables up to stage t. These functions enable us to write the minimization of expected value very succinctly as

$$\min_x f_1\left(x^1\right) + \mathbb{E}\sum_{t=2}^{T} f_t\left(x^t; \vec{x}^{\,t-1}, \vec{\xi}^{\,t}\right) \tag{1}$$

$$\text{subject to } x(\xi) \in X(\xi), \quad \xi \in \varXi, \tag{2}$$

where $x(\xi) = (u(\xi), y(\xi)) \in X(\xi) \subset \mathbb{Z}^k_{\geq 0} \times \mathbb{R}^l$.

In the multi-stage cases of interest to us, $\xi = \left\{\xi^t\right\}_{t=1}^{T}$ is defined on a discrete probability space $(\varXi, \mathcal{A}, \mathcal{P})$. Each scenario, ξ, has probability π_ξ. We organize realizations, ξ, into a tree with the property that scenarios with the same realization up to stage t share a node at that stage. Consequently, $\vec{\xi}^{\,t}$ refers also to a node in the scenario tree. Let $\mathcal{G}_t$ be the set of all scenario tree nodes for stage t and let $\mathcal{G}_t(\xi)$ be the node at time t for a particular scenario, ξ. For a particular node $\mathcal{D}$, let $\mathcal{D}^{-1}$ be the set of scenarios that define the node.

In the presence of a scenario tree, non-anticipativity must be enforced at each non-leaf node, so using the discrete scenario tree notation, problem (1) becomes

$$\min_{x,\hat{x}} \sum_{\xi\in\varXi} \pi_\xi \left[f_1(x^1(\xi)) + \sum_{t=2}^{T} f_t\left(x^t(\xi); \vec{x}^{\,t-1}, \vec{\xi}^{\,t}\right)\right] \tag{3}$$

$$\text{s.t. } x(\xi) \in Y_\xi, \quad \xi \in \varXi \tag{4}$$

$$x^t(\xi) - \hat{x}^t(\mathcal{D}) = 0, \quad t = 1, \ldots, T-1, \ \mathcal{D} \in \mathcal{G}_t, \ \xi \in \mathcal{D}^{-1}. \tag{5}$$

We use the notation $x(\xi)$ to emphasize that the decisions can depend on the realizations of the random variables, while constraints (5) use the auxiliary variable $\hat{x}$ to assure that at every node of the decision tree, the portion of the solution corresponding that decision stage is the same. So while x depends on ξ, it does so in a way that is *non-anticipative*. To put it another way: looking only at expression (3), one might get the impression that the optimization is allowed to be prescient and make use of knowledge of the future in setting stage variable values since x^t in these expressions depends on the entire realization, ξ. However, constraints (5) force the decisions to depend only on information that would be available when they are made. Note that in this formulation, there is one full decision vector x for each scenario. The variables $\hat{x}$ are often called *system of vectors* since they are tied to the tree with partial vectors corresponding to each node of the tree.

Sometimes this problem is written without the variable $\hat{x}$ as follows:

$$\min_{x} \sum_{\xi\in\varXi} \pi_\xi \left[f_1(x^1(\xi)) + \sum_{t=2}^{T} f_t\left(x^t(\xi); \vec{x}^{\,t-1}, \vec{\xi}^{\,t}\right)\right] \tag{6}$$

$$\text{s.t. } x(\xi) \in X(\xi), \quad \xi \in \varXi \tag{7}$$

$$x^t(\xi) - \bar{x}(\mathcal{G}_t(\xi)) = 0, \quad t = 1, \ldots, T-1, \ \xi \in \varXi \tag{8}$$

where for each $t = 1, \ldots, T$ and each $\mathcal{D} \in \mathcal{G}_t$

$$\bar{x}^t(\mathcal{D}) := \sum_{\xi\in\mathcal{D}^{-1}} \pi_\xi x^t(\xi) / \sum_{\xi\in\mathcal{D}^{-1}} \pi_\xi.$$

In other words: $\bar{x}$ is a system of node-by-node averages. Unless there are scenarios with zero probability, the formulation with $\bar{x}$ is equivalent to the formulation with $\hat{x}$, so most practical applications we use the second form and remove any zero probability scenarios in a pre-processing step.

1.2. The PH algorithm

In the context of the formulation specified above, we now present the progressive hedging algorithm for multi-stage stochastic mixed-integer programs:

Algorithm 1 The Progressive Hedging Algorithm for Multi-Stage SMIPs

1. **Initialization:** Let $\nu \leftarrow 0$ and $w_\nu(\mathcal{G}_t(\xi)) \leftarrow 0$, $\forall \xi \in \varXi$, $\forall t \in \{1, \ldots, T\}$. Compute for each $\xi \in \varXi$:
$$x_{\nu+1}(\xi) \in \operatorname*{argmin}_{x\in X(\xi)} \sum_{\xi\in\varXi} \pi_\xi \left[f_1(x^1(\xi)) + \sum_{t=2}^{T} f_t\left(x^t(\xi); \vec{x}^{\,t-1}, \vec{\xi}^{\,t}\right)\right].$$
2. **Iteration Update:** $\nu \leftarrow \nu + 1$.
3. **Aggregation**: Compute for each $t \in \{1, \ldots, T-1\}$ and each $\mathcal{D} \in \mathcal{G}_t$:
$$\bar{x}^t_\nu(\mathcal{D}) \leftarrow \left(\sum_{\hat{\xi}\in\mathcal{D}^{-1}} \pi_{\hat{\xi}} x^t_\nu(\mathcal{G}_t(\xi))\right) / \left(\sum_{\hat{\xi}\in\mathcal{D}^{-1}} \pi_{\hat{\xi}}\right).$$
4. **Weight Update:** Compute for each $t \in \{1, \ldots, T-1\}$ and each $\xi \in \varXi$:
$$w_\nu(\mathcal{G}_t(\xi)) \leftarrow w_{\nu-1}(\mathcal{G}_t(\xi)) + \rho[x^t_\nu(\mathcal{G}(\xi)) - \bar{x}^t_\nu(\mathcal{G}^t(\xi))].$$
5. **Decomposition:** Compute for each $\xi \in \varXi$:
$$x_{\nu+1}(\xi) \in \operatorname*{argmin}_{x\in X(\xi)} f_1(x^1(\xi)) + \sum_{t=2}^{T} f_t\left(x^t(\xi); \vec{x}^{\,t-1}, \vec{\xi}^{\,t}\right) + \sum_{t=1}^{T-1}\left[\langle w^t_\nu(\xi), x^t\rangle + \frac{\rho}{2}\|x^t - \bar{x}^t_\nu(\xi)\|^2\right].$$
6. **Termination criterion:** If the solutions at the tree nodes are equal (up to a given tolerance ϵ) or the maximum iteration count is reached, stop. Otherwise, return to step 2.

We refer to

$$\frac{\rho}{2}\|x^t - \bar{x}^t_\nu(\xi)\|^2$$

as the *proximal term*. Our implementation uses the L_2 norm. We refer to systems of weights w that satisfy

$$\sum_{\xi\in\varXi} \pi(\xi) w(\xi) = 0$$

as *qualified weights*. When PH uses a single value of the parameter ρ for all iterations, scenarios, and variables, we refer to the ρ as a *global* ρ. If PH uses a constant vector ρ for all iterations, we refer to the ρ as a *variable-specific* ρ. If ρ is allowed to change at any iteration, we refer to the ρ as a *dynamic* ρ.