Stochastic Programming
Lecture 10

Dr. Brian Denton

NCSU

October 5, 2010

Today's Class

1. Paper Presentation Assignments
2. LP Solvers
3. Feasibility
4. Bunching

Outline

Paper Presentation Assignments

- Sampling Methods (Group 1: Alex and Mike, Group 2: Clay and Omar), week of Oct 18
- Stochastic Integer Programming (Group 1: Behzad and Shahrzad, Group 2: Hamed and Brian), week of Oct 25
- Multistage Stochastic Programming (Claire and Nils), Nov 2
- Bounding Methods (Yie and Yu), Nov 4
- Lagrangian Methods (Bjorn and Yingying), Nov. 16
- Probabilistic Constraints (Dr. Uzsoy), Nov. 18

Paper Presentation Assignments

Game Plan:

- Identify your paper 2 weeks in advance and send it to the class
- Send me a draft of your slides <u>at least 2 days in advance</u>
- Provide insights including detailed <u>discussion of proofs</u> of the most interesting theorems, <u>an example illustrating the method</u>.
- Class evaluations will be used to determine 50% of your participation grade

Outline

LP Solvers

There are many LP solvers you can use for this course. I recommend one of the following:

- C/C++ and ILOG CPLEX Callable Library (very fast and flexible)
- ILOG OPL (fast and easier to implement some things)
- Matlab (slow but easier to implement)

Outline

Feasibility

Feasibility cuts are added in Step 2 of the L-shaped method. This effort can be avoided if...

Feasibility

Feasibility cuts are added in Step 2 of the L-shaped method. This effort can be avoided if...

- The 2-SLP has *complete* or *relatively complete recourse*
- When *induced constraints* can be derived and added to the first stage to guarantee feasibility

Feasibility

Theorem 2 (B&L Ch 5).

Assume that W is such that $t \in pos(W)$ for all $t \geq 0$. Define $a_i = \min_{k=1,...,K}\{h_{ik}\}$ to be the componentwise minimum of h. Also assume there exists one realization $h_\ell, \ell \in \{1, ..., K\}$ s.t. $a = h_\ell$. Then, $x \in K_2$ if and only if $Wy = a - Tx, y \geq 0$ is feasible.

Outline

Bunching

In step 3 the second stage subproblems must be solved many times to obtain optimal dual solutions, π_k. *Bunching* recognizes many subproblems may have the same optimal basis, B.

$$\pi_k = q_{k,B} B^{-1}, \quad q_k - \pi_k W \geq 0, \quad B^{-1}(h_k - T_k x) \geq 0$$

where $q_{k,B}$ denotes the vector of elements of q_k that correspond to columns of B. Computational savings are achieved if the same basis, B, is optimal for several scenarios k.

Bunching

Bunching is particularly important when q is fixed (why?).

Bunching

Bunching is particularly important when q is fixed (why?).
Assuming fixed q define

$$\tau = \{t \mid t = h_k - T_k x \,, k = 1, ..., K\}$$

as the set of all right hand sides in the second stage problem. Define a bunch, Bu, as:

$$Bu = \{t \in \tau \mid B^{-1} t \geq 0\}$$

Bu is the set of right hand sides that satisfy the feasibility condition. Thus the same π is the optimal dual solution for all $t \in Bu$.

Full Decomposability

For small problems it may be possible to work out a full decomposition of *pos*(W) into component bases. Consider the Farmer Ted problem:

$$
\begin{aligned}
Q(x,\xi) = \min 238y_1 - 170y_2 + 210y_3 - 150y_4 - 36y_5 - 10y_6 \\
s.t. \quad y_1 - y_2 - w_1 = 200 - \xi_1 x_1 \\
y_3 - y_4 - w_2 = 240 - \xi_2 x_2 \\
y_5 + y_6 + w_3 = \xi_3 x_3 \\
y_5 + w_4 = 6000 \\
y, w \geq 0
\end{aligned}
$$

Full Decomposability

The problem has *complete recourse*, i.e., $pos(W) = \Re^4$. The matrix W is

$$W = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

This 4×10 matrix theoretically has $\binom{10}{4} = 210$ bases. However, the LP has the following useful properties:

- y_5 is always in the optimal basis
- w_1, w_2, and w_3 are never in the optimal basis because they are dominated by y_2, y_4, and y_6 respectively
- y_1 or y_2 and y_3 or y_4 are always in the optimal basis

Full Decomposability

Using this information an analytical expression for the dual of $Q(x, \xi)$ can be written as:

$$\pi_1(\xi) = \begin{cases} 238 & \text{if } \xi_1 x_1 < 200 \\ -170 & \text{otherwise} \end{cases}$$

$$\pi_2(\xi) = \begin{cases} 210 & \text{if } \xi_2 x_2 < 240 \\ -150 & \text{otherwise} \end{cases}$$

$$\pi_3(\xi) = \begin{cases} -36 & \text{if } \xi_3 x_3 < 6000 \\ 0 & \text{otherwise} \end{cases}$$

$$\pi_4(\xi) = \begin{cases} 10 & \text{if } \xi_3 x_3 > 6000 \\ 0 & \text{otherwise} \end{cases}$$

The complete decomposition has $2 \times 2 \times 2 \times 2 = 8$ possible optimal bases.

Full Decomposability

It is rare that a recourse problem is fully decomposable. Usually it is only feasible for problems that are

- Very small
- Have a separable second stage LP (e.g. a simple recourse problem)
- Have a very simple structure

Identifying Bunches

Bunching is more generally applicable to 2-SLPs.

For a simple bunching procedure let $\tau = \{t \mid t = h_k - T_k x \text{ for some } k = 1, ..., K\}$ be the set of right-hand sides in the second stage. Proceed as follows:

- Pick some k such that $t_k = h_k - T_k x$
- Let B_1 be the corresponding optimal basis for t_k and $\pi(1)$ the corresponding optimal dual solution
- Let $Bu(1) = \{t \in \tau \mid B_1^{-1} t \geq 0\}$
- Let $\tau_1 = \tau \setminus Bu(1)$
- Repeat, i.e, choose some element of τ_1 and set $Bu(2) = \{t \in \tau_1 \mid B_2^{-1} t \geq 0\}$ and $\tau_2 = \tau_1 \setminus Bu(2)$, and so on....

Identifying Bunches

Given b bunches, the optimality cuts can be written as:

$$E_{s+1} = \sum_{\ell=1}^{b} \pi(\ell) \sum_{t_k \in Bu(\ell)} p_k T_k$$

$$e_{s+1} = \sum_{\ell=1}^{b} \pi(\ell) \sum_{t_k \in Bu(\ell)} p_k h_k$$

Bunching has some drawbacks:

- Bunches change from one iteration of L-shaped method to another
- The same t_k may be checked against many different bases
- A new linear program must be solved for every bunch

See B & L, p. 173 for an example.

Identifying Bunches

For next time read about:

- Section 5.5 – Inner Linearization (a.k.a. Dantzig-Wolfe Decomposition) and its relationship to the L-shaped method
- Section 5.6 – Basis Factorization Method
- Section 5.7 – Special Cases: Simple Recourse and Network Flow Problems