Stochastic Programming using Algebraic Modeling Languages

Timo Lohmann, Ronny Hansmann, TU Braunschweig
Michael Bussieck, GAMS Software GmbH

1. September 2010
Agenda

- Short SP Introduction
- What Algebraic Modeling Languages offer
- Solving SPs: Decomposition Algorithms
- An Example
- Rapid Algorithm Development in GAMS
Agenda

- Short SP Introduction
- What Algebraic Modeling Languages offer
- Solving SPs: Decomposition Algorithms
- An Example
- Rapid Algorithm Development in GAMS
Introduction

- Starting with a deterministic problem:

\[
\begin{align*}
\min & \quad g_0(x) \\
\text{s.t.} & \quad g_i(x) \leq 0, \quad i = 1, \ldots, m
\end{align*}
\]

\[x \in X \subset \mathbb{R}^n\]

- Problem can be linear, nonlinear (convex, nonconvex) depending on the functions \(g_i\) and the set \(X\).

- Adding uncertainty:

\[
\begin{align*}
\min & \quad g_0(x, \xi) \\
\text{s.t.} & \quad g_i(x, \xi) \leq 0, \quad i = 1, \ldots, m
\end{align*}
\]

\[x \in X \subset \mathbb{R}^n\]

where \(\xi\) is a random vector varying over the set \(\Xi \subset \mathbb{R}^k\)
Multi-stage decision making

- A stage is characterized by new information becoming known at the beginning of the stage and making recourse decisions / adjustments at the end of the stage (stage 1 is an exception):

  Make decision → Observe uncertainty → Make adjustment

  Time

  Stage 1  Stage 2  Stage 3  ...  Stage T

- When making decisions only outcomes of the current stage and previous stages are available. For future stages only expectations exist.

  -> Non-anticipativity of the stochastic process

Source: Fourer and Lopes (2009)
Multi-stage model

- It is often assumed that the decision in the current stage only depends on the previous stage. The multi-stage model then reads:

\[
\min c_1 x_1 + E[\min c_2 \xi^2 x_2 \xi^2 + \cdots + E[\min c_T (\xi^T) x_T (\xi^T)]] \cdots
\]

\[\text{s.t. } W_1 x_1 = h_1\]

\[T_1 (\xi^2) x_1 + W_2 (\xi^2) x_2 (\xi^2) = h_2 (\xi^2)\]

\[\vdots\]

\[T_{T-1} (\xi^T) x_{T-1} (\xi^{T-1}) + W_T (\xi^T) x_T (\xi^T) = h_T (\xi^T)\]

\[x_1 \geq 0, \ x_t (\xi^t) \geq 0, \ t = 2, \ldots, T.\]

- This assumption benefits decomposition algorithms because they can exploit the structure of the coefficients matrix.

- Another common assumption is that the recourse matrices $W$ are fixed (fixed recourse) or have a structure which eliminates the possibility of infeasibility (complete recourse).
Agenda

- Short SP Introduction
- What Algebraic Modeling Languages offer
- Solving SPs: Decomposition Algorithms
- An Example
- Rapid Algorithm Development in GAMS
Algebraic modeling languages

- Tested modeling languages:

[Logos of AIMMS, AMPL, GAMS, LINGO, Maximal]
What algebraic modeling languages offer

- Features for stochastic programming:
  - Separation of deterministic core model and stochastic information
  - Ways of describing stochastic information:
    - Scenarios, either explicit provided by the user or constructed by the software (sampling techniques)
    - Continuous and discrete distributions (with intra- and interstage correlations)
  - Support for multi-stage and two-stage modeling, e.g. automatic assignment of variables and equations to stages
  - Scenario tree construction support, e.g. visualization and reduction methods
  - Decomposition algorithms, e.g. stochastic version of Benders' decomposition
  - Solution reports featuring
    - Solutions for each scenario
    - Value of the stochastic solution, expected value of perfect information
  - Reading from and writing into other formats, such as SMPS
The most important features

- Features for stochastic programming:
  - **Separation** of deterministic core model and stochastic information
  - Ways of describing stochastic information:
    - **Scenarios**, either explicitly provided by the user or constructed by the software (sampling techniques)
    - Continuous and discrete distributions (with intra- and interstage correlations)
  - Support for multi-stage and two-stage modeling, e.g. **automatic assignment** of variables and equations to stages
  - Scenario tree construction support, e.g. visualization and reduction methods
  - **Decomposition algorithms**, e.g. stochastic version of Benders’ decomposition
  - Solution reports featuring
    - Solutions for each scenario
    - **Value of the stochastic solution**, expected value of perfect information
  - Reading from and writing into other formats, such as SMPS
Agenda

- Short SP Introduction
- What Algebraic Modeling Languages offer
- Solving SPs: Decomposition Algorithms
- An Example
- Rapid Algorithm Development in GAMS
Basic idea

- Reconsider the multi-stage problem:

\[
\begin{align*}
\min & \quad c_1 x_1 + E[ \min \ c_2 \xi^2 x_2 \xi^2 + \cdots + E[ \min \ c_T(\xi^T) x_T(\xi^T) ] \cdots ] \\
\text{s.t.} & \quad W_1 x_1 = h_1 \\
& \quad T_1(\xi^2)x_1 + W_2(\xi^2)x_2(\xi^2) = h_2(\xi^2) \\
& \quad \vdots \\
& \quad T_{T-1}(\xi^{T-1})x_{T-1}(\xi^{T-1}) + W_T(\xi^T)x_T(\xi^T) = h_T(\xi^T) \\
& \quad x_1 \geq 0, \quad x_t(\xi^t) \geq 0, \quad t = 2, \ldots, T.
\end{align*}
\]

- With more stages and more scenarios the size explodes and the DE problem cannot be solved in a reasonable amount of time.

- Idea: Only look at the current stage and aggregate the future stages into a function depending on the current stage decision.

-> The Future Cost Function
One-stage subproblems

- Define one-stage dispatch subproblems:
  \[
  \min c_t x_t + \hat{\alpha}_{t+1}(x_t)
  \]
  \[
  \text{s.t. } W_t x_t \geq h_t - T_{t-1} x^*_{t-1}.
  \]

- Note that variables \( x^*_{t-1} \) are fixed. Instead of solving the large multi-stage problem we solve a sequence of one-stage problems.
- Question: How to construct the future cost function(s) \( \hat{\alpha}_{t+1}(x_t) \)?
- Possible approaches:
  - SDP (Stochastic Dynamic Programming): A state-space approach using a recursion scheme and interpolations between obtained solution points
  - SDDP (Stochastic Dual Dynamic Programming) / Benders‘: Construct a piecewise linear function using supporting hyperplanes (cuts) through recursion.
Construction of cuts (DDP)

- Solve the last stage subproblem (t=T):
  \[ \min c_t x_t + \hat{\alpha}_{t+1}(x_t) \]
  \[ \text{s.t. } W_t x_t \geq h_t - T_{t-1} x_{t-1}^* \]

- Calculate marginals \( \pi^{1\ T}_{T-1} \) of the constraints and use them as dual multipliers for stage T-1:
  \[ \hat{\alpha}_{T-1}(x_{T-2}^*) = \min c_{T-1} x_{T-1} + \hat{\alpha}_T \]
  \[ \text{s.t. } W_{T-1} x_{T-1} \geq h_{T-1} - T_{T-2} x_{T-2}^* \]
  \[ \hat{\alpha}_T \geq \pi^{1\ T}_{T}(h_T - T_{T-1} x_{T-1}) \]

- When calculating marginals for earlier stages the marginals for the cuts have to be accounted for as well (e.g. stage T-2):
  \[ \hat{\alpha}_{T-1} \geq \pi^{1\ T-1}_{T-1}(h_{T-1} - T_{T-2} x_{T-2}) + \lambda^{1\ T-1}_{T-1} \pi^{1\ T-1}_{T-1} h_T \]

Source: Pereira and Pinto (1991)
Construction of cuts (DDP)

- The complete one-stage subproblem with cuts reads:

\[
\hat{\alpha}_t(x^*_{t-1}) = \min \ c^T_t x_t + \hat{\alpha}_{t+1} \\
\text{s.t. } W_t x_t \geq h_t - T_{t-1} x^*_{t-1} \\
\hat{\alpha}_{t+1} + \pi_{t+1}^j T_t x_t \geq \delta^j_t, \ j = 1, \ldots, J,
\]

\[
\delta^j_t = \begin{cases} 
\pi_{t+1}^j h_{t+1}, & t = T - 1 \\
\pi_{t+1}^j h_{t+1} + \sum_{i=1}^{j} \lambda_{i,t+1}^j \delta^i_{t+1}, & t = 1, \ldots, T - 2,
\end{cases}
\]

\[
\tag{1}
\]

Source: Pereira and Pinto (1991)
Construction of cuts (SDDP)

- Initialize:
  
  Let $T$ be the planning horizon, initialize $\alpha_{t+1}(x_t) = 0$ for $t = 1, \ldots, T$; iteration count $J=0$; Define a set of trial solutions $\{x_{tn}^*, n = 1, \ldots, N, t = 1, \ldots, T\}$.

- Carry out a Backward Recursion:

  Backward recursion. Repeat for $t = T, \ldots, 2$ :

  Repeat for each trial decision $x_{t,n}^*$, $n = 1, \ldots, N$:

  Repeat for each realization $h_{t,m}$, $m = 1, \ldots, M$:

  Solve problem (1) using trial decision $x_{t-1,n}^*$. Let $\pi_{t,m}^j$ and $\lambda_{i,t,m}^j$ be the multipliers associated to the constraints (1a) and (1b), respectively.

  Calculate the expected vertex value $\overline{\pi}_{t,n}^j = \sum_{m=1}^{M} p_{t,m} \pi_{t,m}^j$ and $\overline{\delta}_{t-1,n}^j = \sum_{m=1}^{M} p_{t,m} \delta_{t-1,m}^j$, and construct one supporting hyperplane of the approximate expected future cost function for stage $t-1$, $\overline{\alpha}_t(x_{t-1})$.

- Solve the first-stage problem (1) for $t=1$, update $x_{1,n}^* := x_1^*$

Source: Pereira and Pinto (1991)
Building a new solution and check for convergence (SDDP)

- Carry out a forward simulation:

  Forward Simulation. Repeat for $t = 2, \ldots, T$:

  Repeat for $n = 1, \ldots, N$:

  Sample a vector $h_{tn}$ from the set $\{h_{tm}, m = 1, \ldots, M\}$.

  Solve the two-stage subproblem (1) using trial decision $x^*_{t-1,n}$. Use the optimal solution to update $x^*_{tn}$.

- Check for convergence:
  - Update the upper bound: $\bar{z} = c_1 x^*_1 + \frac{1}{N} \sum_{n=1}^{N} \bar{z}_n$
  - Update the lower bound: $\underline{z} = c_1 x^*_1 + \alpha_2(x_1)$.
  - If the difference between the upper confidence bound $\bar{z} + z_{q/2} \sigma_z / \sqrt{M}$ and the lower bound $\underline{z}$ is less than prescribed accuracy level: STOP

Source: Pereira and Pinto (1991)

Source: Shapiro (2009)
Agenda

- Short SP Introduction
- What Algebraic Modeling Languages offer
- Solving SPs: Decomposition Algorithms
- An Example
- Rapid Algorithm Development in GAMS
The hydro power example

- 4 stages, right-hand side random (inflow), discrete distribution (64 scenarios)
- 4 reservoirs for hydro power generation
- Thermal power generation possible but expensive
- Demand L for power has to be satisfied (violation even more expensive)
- Water can be used in successive reservoirs.
- Decisions to make:
  - How much water should be stored and how much water should be used for power generation at each reservoir in each stage?

Source: Pereira and Pinto (1985)

Source: Velasquez, Restrepo, and Campo (1999)
The model formulation

\[
\begin{align*}
\min & \quad \sum_{t=1}^{4} (c \cdot G_T + f \cdot D_t) \\
\text{s.t.} & \quad G_T + \sum_{i=1}^{4} r_i Q_{i,t} + D_t = L \quad \forall t \\
& \quad V_{i,t-1} + a_{i,t} + \sum_{j \in J_i} Q_{j,t} = V_{i,t} + Q_{i,t} \quad \forall i, t \\
& \quad Q_{i,t} \leq Q \quad \forall i, t \\
& \quad G_T \leq \overline{G_T} \quad \forall t \\
& \quad G_T, D_t, Q_{i,t}, V_{i,t} \geq 0 \quad \forall i, t.
\end{align*}
\]

\(G_T\) Thermal power generation

\(D_t\) Unsatisfied demand

\(Q_{i,t}\) Water release for hydro power generation

\(V_{i,t}\) Water volume stored in reservoir i at the end of stage t

Source: Velasquez, Restrepo, and Campo (1999)
Agenda

- Short SP Introduction
- What Algebraic Modeling Languages offer
- Solving SPs: Decomposition Algorithms
- An Example
- Rapid Algorithm Development in GAMS
SDDP in GAMS

- **Equations** can be used in multiple **Model** statements
  - Core model:  
    ```gams
    Model hydro /obj,cont,maxflow,therm,dem/;
    ```
  - Submodels:  
    ```gams
    Model hydrospp / hydro, obj_approx, cuts /
    ```
- **Equation** generation controlled by **subsets**:
  ```gams
  cont(i,tt(t)).. V(i,t-1) + V0(i)$sameas('t1',t) + A(i,t) - Q(i,t) 
    + sum(M(i,ii), Q(ii,t)) =e= V(i,t);
  cuts(jj,n,tt(t))$docuts..
  ALPHA(t+1) - sum(i, cont_m(jj,n,i,t+1)*V(i,t)) =g= delta(jj,n,t);
  ```
- **Loops** over **sets**:
  ```gams
  loop(s, // for all realizations
    A(i,tloop)=Astoch(s,i,tloop);
  )
  ```
- Change **parameters** of a model
- Access of model stats for calculations:
  ```gams
  delta(j,nloop,tloop-1) = 
      1/card(s)*[sum(i,-cont.m(i,tloop)*A(i,tloop) 
        + maxflow.m(i,tloop)*Qcap(i)) 
        + therm.m(tloop)*GTCap + dem.m(tloop)*L 
        + docuts*sum((jj,n), cuts.m(jj,n,tloop)*delta(jj,n,tloop))] 
        + delta(j,nloop,tloop-1);
  ```
Solver integration

- Running the SDDP algorithm with default GAMS settings is slow because „communicating“ with the solver for the small one-stage dispatch models takes up most of the time.
- „Communicating“ here means:
  - Model generation in GAMS
  - Writing the model to the hard drive (GAMS vacates memory)
  - Starting the solver
  - Restart GAMS; swap of GAMS database
- By using `hydrosp.solvelink = %solvelink.LoadLibrary%` the solver DLL is used in the GAMS process which means
  - GAMS stays in memory, no swap of GAMS database
  - Fast memory based model communication
Scenario solver and comparison

- Another speed improvement is possible by using the GAMS scenario solver (beta state). Advantage: GAMS generates the model once and the scenario solver replaces the parameters accordingly for each solve.

  \texttt{Solve} hydros\texttt{p} min ACOST using LP scenario dict;

<table>
<thead>
<tr>
<th>Setting</th>
<th>Solve time (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solvelink=0 \textit{(default)}</td>
<td>40.297</td>
</tr>
<tr>
<td>Solvelink=%Solvelink.LoadLibrary%</td>
<td>03.625</td>
</tr>
<tr>
<td>Scenario Solver</td>
<td>00.797</td>
</tr>
</tbody>
</table>

(GAMS 23.5, lp=cplexd, Intel Core2 @ 2.0 GHz, 2GB)

- Using the scenario solver has to be done via the „Scenario Dict“