TWO STAGE STOCHASTIC LINEAR PROGRAMMING WITH GAMS

ERWIN KALVELAGEN

Abstract. This document shows how to model two-stage stochastic linear programming problems in a GAMS environment. We will demonstrate using a small example, how GAMS can be used to formulate and solve this model as a large LP or using specialized stochastic solvers such as OSL-SE and DECIS. Finally a tailored implementation of the Benders Decomposition algorithm written in GAMS is used to solve the model.

1. Introduction

Stochastic programming has become an important problem area. With current standard off-the-shelf software including modeling systems such as AMPL and GAMS, powerful large-scale general-purpose solvers such as Cplex and specialized stochastic programming solvers such as OSL-SE and DECIS, end-users can develop realistic stochastic programming models and solve them on standard desktop hardware.

2. Two-stage stochastic linear programming problems

The two-stage stochastic linear programming problem can be stated as [2, 5, 8]:

SLP	$\underset{x}{\text{minimize}}$	$c^T x + E_\omega Q(x,\omega)$
		$Ax = b$
		$x \geq 0$

where

$$
\begin{aligned}
Q(x,\omega) = \min_y \; & d_\omega^T y \\
& T_\omega x + W_\omega y = h_\omega \\
& y \geq 0
\end{aligned}
\tag{1}
$$

Here E_ω is the expectation, and ω denotes a scenario or possible outcome with respect to the probability space (Ω, P). The variables x are called first-stage variables, as they have to be decided upon before the outcome of the stochastic variable ω is observed. The variables y are second-stage variables: they can be calculated after the outcome of ω is known.

We will consider discrete distributions P only, so we can write:

$$
E_\omega Q(x,\omega) = \sum_{\omega \in \Omega} p(\omega) Q(x,\omega) \tag{2}
$$

Date: 8 january 2003.

Using this we can formulate a large LP that forms the *deterministic equivalent* problem:

$$\begin{aligned} \min\ & c^T x + \sum_{\omega} p(\omega) d_\omega^T y_\omega \\ & Ax = b \\ & T_\omega x + W_\omega y_\omega = h_\omega \ \forall \omega \\ & x \ge 0, y_\omega \ge 0 \end{aligned} \tag{3}$$

The chain of events in this model is as follows: first the decision maker implements the first stage decisions x. Then the system will be subjected to the random process described by (Ω, P), which results in an outcome $\omega \in \Omega$. Finally the decision maker will execute the second stage decisions y accordingly.

3. Example

The standard transportation model can be stated as:

TRANSPORT	$\underset{x}{\text{minimize}}\ \sum_{i,j} c_{i,j} x_{i,j}$
	$\sum_j x_{i,j} = s_i \ \forall i$
	$\sum_i x_{i,j} = d_j \ \forall j$
	$x_{i,j} \ge 0$

where $c_{i,j}$ are the unit transportation costs, s_i is the supply at plant i and d_j is the demand at location j.

Now consider the case that demand d_j is stochastic. We assume that products are shipped before the actual demand is observed. After the products arrive at each location j, actual demand is known. If demand is not met, the sales is lost. If the shipment is larger than the final demand, then the remaining products need to be disposed of as they have no shelf life (i.e. they spoil). Unit disposal costs are known.

To model this, first assume we somehow know the demand. This means we can build a non-stochastic version of the model, also called the *core model*:

$$\begin{aligned} \max\ & \sum_j p_j Sales_j - \sum_{i,j} c_{i,j} Ship_{i,j} - \sum_i c_i Prod_i - \sum_j c_j Waste_j \\ & Prod_i = \sum_j Ship_{i,j} \\ & Prod_i \le cap_i \\ & \sum_i Ship_{i,j} = Sales_j + Waste_j \\ & Sales_j \le demand_j \\ & Sales_j \ge 0, Ship_{i,j} \ge 0, Prod_i \ge 0, Waste_j \ge 0 \end{aligned} \tag{4}$$

In this model we try to maximize profit, i.e. revenue minus costs, where costs can be broken down in production costs, transportation costs and waste disposal costs.

	Outcome			Probability		
j	Lo	Mid	Hi	Lo	Mid	Hi
1	150	160	170	.25	.50	.25
2	100	120	135	.25	.50	.25
3	250	270	300	.25	.50	.25
4	300	325	350	.30	.40	.30
5	600	700	800	.30	.40	.30

TABLE 1. Probability distribution of demand figures

This model is just a little bit more complex than the transportation. When solved like this, it is expected the model will choose $Waste_j = 0$ as it does not contribute to the profit. When we substitute this value in the model, the transportation model follows immediately.

To quickly assess the correctness of the model one can solve a few instances of this model. We need to have some values for the demand. If only distributions are known some good candidates are setting $demand_j$ to the mean, the worst or best possible cases, or using some random drawings.

When we make $demand_j$ stochastic, the problem becomes more involved. First we need to define the probability distribution. Assume the data from table 1 are used.

We assume the probabilities are independent. To calculate the joint probabilities for all demand markets, we note that

$$\begin{aligned} Pr(d_1 = 150 \wedge d_2 = 100 \wedge d_3 = 250 \wedge d_4 = 300 \wedge d_5 = 600) = \\ .25 \times .25 \times .25 \times .30 \times .30 = 0.00140625 \end{aligned} \tag{5}$$

There are in total $3^5 = 243$ *scenario's*.

We are now able to formulate the deterministic equivalent of the stochastic model:

$$\begin{aligned} \max Profit = & \sum_{j,\omega} p_j prob_\omega Sales_{j,\omega} - \sum_{i,j} c_{i,j} Ship_{i,j} \\ & - \sum_i c_i Prod_i - \sum_{j,\omega} c_j prob_\omega Waste_{j,\omega} \\ & Prod_i = \sum_j Ship_{i,j} \\ & Prod_i \leq cap_i \\ & \sum_i Ship_{i,j} = Sales_{j,\omega} + Waste_{j,\omega} \\ & Sales_{j,\omega} \leq demand_j \\ & Sales_{j,\omega} \geq 0, Ship_{i,j} \geq 0, Prod_i \geq 0, Waste_{j,\omega} \geq 0 \end{aligned} \tag{6}$$

It is noted that the constraint

$$\sum_i Ship_{i,j} = Sales_{j,\omega} + Waste_{j,\omega} \tag{7}$$

is not very efficient as stated: the term $\sum_i Ship_{i,j}$ is repeated for all possible values of ω. In a case like this it is better to introduce a modest amount of extra intermediate variables and equations in order to achieve a significant reduction in the number of nonzero elements. We therefore would prefer the following formulation:

$$\begin{aligned} Received_j &= \sum_i Ship_{i,j} \\ Received_j &= Sales_{j,\omega} + Waste_{j,\omega} \end{aligned} \tag{8}$$

A few remarks can be made. The second stage is only how to deal with the products received: they are either sold (up to $demand_j$) or disposed of. This is a particular easy decision: sell as much as you can, and destroy the rest. In more complicated situations, after an observation $\omega \in \Omega$ becomes available, we need either to solve a (smaller) second stage LP model or in case we recorded all second stage variables, we can select the correct solution values. It is noted that the second stage problem is feasible for any $\omega \in \Omega$.

In this example we assumes the events for each demand market were independent. This may not be a valid assumption: if the market region j is down then it may be more likely that market region k is also likely to suffer. In case of perfect correlation, we would end up with a much smaller model as there would be only three scenario's: low, mid and high.

The complete model formulated in GAMS is reproduced below.

Model stochdeteq.gms. [1]

```
$ontext

   Transportation problem with stochastic demands.

   Erwin Kalvelagen, January 2003


$offtext

sets
  i 'factories' /f1*f3/
  j 'distribution centers' /d1*d5/
  s 'individual scenarios' /lo,mid,hi/
;

parameter capacity(i) /f1 500, f2 450, f3 650/;

table demand(j,s) 'possible outcomes for demand'
      lo  mid   hi
  d1 150  160  170
  d2 100  120  135
  d3 250  270  300
  d4 300  325  350
  d5 600  700  800
;

table prob(j,s) 'probabilities for table demand'
      lo  mid   hi
  d1 .25  .50  .25
  d2 .25  .50  .25
  d3 .25  .50  .25
  d4 .30  .40  .30
  d5 .30  .40  .30
;

loop(j, abort$(abs(sum(s,prob(j,s))-1)>0.001) "probabilities don't add up");
```

[1] http://www.amsterdamoptimization.com/models/twostage/stochdeteq.gms

```
*
* set up joint probabilities
* total scenarios =  3**5 = 243
*
set js 'scenarios for joint probabilities' /js1*js243/;
parameter jdemand(j,js) 'joint distribution: outcomes';
parameter jprob(js)   'joint distribution: probabilities';

alias (s,s1,s2,s3,s4,s5);
set current(js);
current('js1') = yes;
loop((s1,s2,s3,s4,s5),
   jdemand('d1',current) = demand('d1',s1);
   jdemand('d2',current) = demand('d2',s2);
   jdemand('d3',current) = demand('d3',s3);
   jdemand('d4',current) = demand('d4',s4);
   jdemand('d5',current) = demand('d5',s5);
   jprob(current) = prob('d1',s1)*prob('d2',s2)*prob('d3',s3)*prob('d4',s4)*prob('d5',s5);
   current(js) = current(js-1);
);
display jdemand,jprob;

scalar sumprob;
sumprob = sum(js, jprob(js));
display sumprob;
abort$(abs(sumprob-1)>0.00001) "joint probabilities don't add up";

table transcost(i,j) 'unit transportation cost'
       d1    d2    d3    d4    d5
  f1   2.49  5.21  3.76  4.85  2.07
  f2   1.46  2.54  1.83  1.86  4.76
  f3   3.26  3.08  2.60  3.76  4.45
;

scalar prodcost  'unit production cost' /14/;
scalar price     'sales price' /24/;
scalar wastecost 'cost of removal of overstocked products' /4/;


*---------------------------------------------------------------------
* first we formulate a non-stochastic version of the model
* we just use 'mid' values for the demand.
*---------------------------------------------------------------------

variables
   ship(i,j)   'shipments'
   product(i)  'production'
   sales(j)    'sales (actually sold)'
   waste(j)    'overstocked products'
   profit
;
positive variables ship,product,sales,waste;

equations
   obj
   production(i)
   selling(j)
;


obj.. profit =e= sum(j, price*sales(j)) - sum((i,j), transcost(i,j)*ship(i,j))
                 - sum(j, wastecost*waste(j)) - sum(i,prodcost*product(i));


production(i).. product(i) =e= sum(j, ship(i,j));
product.up(i) = capacity(i);

selling(j).. sum(i, ship(i,j)) =e= sales(j)+waste(j);
sales.up(j) = demand(j,'mid');
```

model	Size $m/n/nz$	Obj
nonstoch	9/29/72	11852.30
stoch	1224/2454/6132	10793.00

TABLE 2. Results

```
model nonstoch /obj,production,selling/;
solve nonstoch maximizing profit using lp;

display ship.l,product.l,sales.l,waste.l;
parameter shipnonstoch(i,j);
shipnonstoch(i,j) = ship.l(i,j);

*---------------------------------------------------------------------------
* now we formulate a stochastic version of the model
* We form here the deterministic equivalent
*---------------------------------------------------------------------------

variables
   salesw(j,js) 'stochastic version of sales'
   wastew(j,js)  'stochastic version of waste'
   received(j)  'amount of product received in distribution center'
;
positive variable salesw,wastew;

equations
   objw
   sellingw(j,js)
   receive(j)
;

objw.. profit =e= sum((j,js),price*jprob(js)*salesw(j,js))
                - sum((i,j), transcost(i,j)*ship(i,j))
                - sum((j,js), wastecost*jprob(js)*wastew(j,js))
                - sum(i,prodcost*product(i));

receive(j)..     received(j) =e= sum(i, ship(i,j));
sellingw(j,js).. received(j) =e= salesw(j,js)+wastew(j,js);
salesw.up(j,js) = jdemand(j,js);

model stoch /objw,production,receive,sellingw/;
solve stoch maximizing profit using lp;

display ship.l,product.l,salesw.l,wastew.l;


*---------------------------------------------------------------------------
* Compare with nonstochastic solution
*---------------------------------------------------------------------------

ship.fx(i,j) = shipnonstoch(i,j);
solve stoch maximizing profit using lp;
```

Table 2 shows the sizes and optimal objective values of the two models: the core model using *Mid* observations and the deterministic equivalent formulation of the stochastic model. The objective of the stochastic model needs to be interpreted as the expected profit or in other words the long run average profit if the distribution plan given by the variables $Ship_{i,j}$ is implemented. If we would have used the solution found in the non-stochastic core model, then the average profit would have dropped to 10452.30.

4. A DECIS formulation

The DECIS system [6, 7] for two-stage stochastic linear programming problems is tailored for models with a huge amount of scenario's. The model specification consists of the core model plus separate files with the scenarios. There is no need to generate a large deterministic equivalent model. Inside DECIS advanced statistical sampling techniques are used to evaluate only a subset of all possible scenarios [5], delivering a confidence interval which is sufficiently small.

In order to specify a model for DECIS we first need a core model. For our example we already have such a formulation available in equation (4). The other ingredients of a DECIS formulation are a stage 1/stage 2 classification of all variables and equations and a specification of the probability distributions. For this model, the variables *Ship* are first stage, and the variables *Sales* and *Waste* are second stage variables. The equations with only first stage variables are called first stage equations, and equations that include second stage variables are second stage equations. The objective does not need to be classified.

The probability distributions are specified in a so-called `STOCH` file. This file follows the SMPS notation [1, 4], however instead of MPS names we use GAMS names. In this case we deal with stochastic upperbounds on the variable *Sales*. The syntax for the SMPS file for this situation would be:

```
NAME             GAMS
INDEP         DISCRETE
 UP BND       C0000019   150.0000000   PERIOD2    0.250000000
 UP BND       C0000019   160.0000000   PERIOD2    0.500000000
 UP BND       C0000019   170.0000000   PERIOD2    0.250000000
 UP BND       C0000020   100.0000000   PERIOD2    0.250000000
 UP BND       C0000020   120.0000000   PERIOD2    0.500000000
 UP BND       C0000020   135.0000000   PERIOD2    0.250000000
 UP BND       C0000021   250.0000000   PERIOD2    0.250000000
 UP BND       C0000021   270.0000000   PERIOD2    0.500000000
 UP BND       C0000021   300.0000000   PERIOD2    0.250000000
 UP BND       C0000022   300.0000000   PERIOD2    0.300000000
 UP BND       C0000022   325.0000000   PERIOD2    0.400000000
 UP BND       C0000022   350.0000000   PERIOD2    0.300000000
 UP BND       C0000023   600.0000000   PERIOD2    0.300000000
 UP BND       C0000023   700.0000000   PERIOD2    0.400000000
 UP BND       C0000023   800.0000000   PERIOD2    0.300000000
ENDATA
```

The third column is the column or variable name. Following is the possible outcome. `PERIOD2` is a placeholder, and the last column is the probability. The GAMS/DECIS preprocessor can take a file with GAMS names and translate it correctly to the above format. This file is called `model.stg`:

```
INDEP DISCRETE
UP BND sales d1               150.00 PERIOD2       0.25
UP BND sales d1               160.00 PERIOD2       0.50
UP BND sales d1               170.00 PERIOD2       0.25
UP BND sales d2               100.00 PERIOD2       0.25
UP BND sales d2               120.00 PERIOD2       0.50
UP BND sales d2               135.00 PERIOD2       0.25
UP BND sales d3               250.00 PERIOD2       0.25
UP BND sales d3               270.00 PERIOD2       0.50
UP BND sales d3               300.00 PERIOD2       0.25
UP BND sales d4               300.00 PERIOD2       0.30
UP BND sales d4               325.00 PERIOD2       0.40
UP BND sales d4               350.00 PERIOD2       0.30
UP BND sales d5               600.00 PERIOD2       0.30
UP BND sales d5               700.00 PERIOD2       0.40
UP BND sales d5               800.00 PERIOD2       0.30
```

This file is free format: the column positions are not significant, opposed to the SMPS stoch file. Such a file is of course easily generated by GAMS using the PUT statement.

The SMPS *core file* is generated based on the *core model* formulated in GAMS and the SMPS *time file* is also automatically generated using the stage 1/stage 2 classification. It may be instructive to have a look at these files. After a DECIS run, these files are available under the names `MODEL.COR`, `MODEL.TIM` and `MODEL.STO` is the SMPS *stoch file*.

The complete DECIS model looks like:

Model stochdecis.gms. [2]

```
$ontext

   Transportation problem with stochastic demands.
   DECIS formulation

   Erwin Kalvelagen, January 2003


$offtext

sets
  i 'factories' /f1*f3/
  j 'distribution centers' /d1*d5/
  s 'individual scenarios' /lo,mid,hi/
;

parameter capacity(i) /f1 500, f2 450, f3 650/;

table demand(j,s) 'possible outcomes for demand'
      lo  mid   hi
  d1 150  160  170
  d2 100  120  135
  d3 250  270  300
  d4 300  325  350
  d5 600  700  800
;

table prob(j,s) 'probabilities for table demand'
      lo  mid   hi
  d1 .25  .50  .25
  d2 .25  .50  .25
  d3 .25  .50  .25
  d4 .30  .40  .30
  d5 .30  .40  .30
;

loop(j, abort$(abs(sum(s,prob(j,s))-1)>0.001) "probabilities don't add up");



table transcost(i,j) 'unit transportation cost'
       d1    d2    d3    d4    d5
  f1   2.49  5.21  3.76  4.85  2.07
  f2   1.46  2.54  1.83  1.86  4.76
  f3   3.26  3.08  2.60  3.76  4.45
;

scalar prodcost  'unit production cost' /14/;
scalar price     'sales price' /24/;
scalar wastecost 'cost of removal of overstocked products' /4/;


*---------------------------------------------------------------------
* CORE MODEL
```

[2] http://www.amsterdamoptimization.com/models/twostage/stochdecis.gms

```
*---------------------------------------------------------------------

variables
   ship(i,j)   'shipments'
   product(i)  'production'
   sales(j)    'sales (actually sold)'
   waste(j)    'overstocked products'
   profit
;
positive variables ship,product,sales,waste;

equations
   obj
   production(i)
   selling(j)
;


obj.. profit =e= sum(j, price*sales(j)) - sum((i,j), transcost(i,j)*ship(i,j))
                 - sum(j, wastecost*waste(j)) - sum(i,prodcost*product(i));


production(i).. product(i) =e= sum(j, ship(i,j));
product.up(i) = capacity(i);

selling(j).. sum(i, ship(i,j)) =e= sales(j)+waste(j);
sales.up(j) = demand(j,'mid');


*---------------------------------------------------------------------
* STAGE 1/STAGE 2 CLASSIFICATION
*---------------------------------------------------------------------

*
* variables
*
ship.stage(i,j) = 1;
waste.stage(j) = 2;
sales.stage(j) = 2;

*
* equations
*
production.stage(i) = 1;
selling.stage(j) = 2;

*---------------------------------------------------------------------
* WRITING THE .STG FILE
*---------------------------------------------------------------------

file stg /model.stg/;
put stg;

put "INDEP DISCRETE"/;
loop(j,
    loop(s,
         put "UP BND sales ",j.tl,demand(j,s)," PERIOD2",prob(j,s)/;
    );
);
putclose stg;


*---------------------------------------------------------------------
* output a MINOS option file
*---------------------------------------------------------------------

file mopt / MINOS.SPC /;
put mopt;
put "begin"/;
put "rows 250"/;
put "columns 250"/;
put "elements 10000"/;
```

```
put "end"/;
putclose;

*---------------------------------------------------------------------
* Solve with default settings
*---------------------------------------------------------------------


option lp=decism;
model m /obj,production,selling/;
solve m maximizing profit using lp;


*---------------------------------------------------------------------
* Solve exactly using DECIS option ISTRAT 4
*---------------------------------------------------------------------


file decopt / decism.opt /;
put decopt;
put '4 "ISTRAT"'/;
putclose;

m.optfile=1;
solve m maximizing profit using lp;
```

The first model uses the sampling techniques described earlier. This gives an objective of 10785.00. When we instruct DECIS to calculate an exact solution, by solving the universe problem, the objective is 10793.00 which is identical to our deterministic equivalent formulation.

Notice that we only needed to formulate the simpler core model. There was no need to construct scenario's and calculate joint probabilities. The power of DECIS is being able to model and solve very large problems with extremely large numbers of scenario's.

5. An OSL-SE formulation

OSL-SE is using a different approach[3]. The formulation is based on the deterministic equivalent. A disadvantage is that GAMS has to handle a possibly very large model, may be even too large to generate comfortably. An obvious advantage is that the model can be verified simply by using a standard LP solver.

OSL-SE is solver targeted for multi-stage problems and therefore the GAMS formulated is organized around the notion of a scenario tree. In our case the tree has a root, and then 243 leaf nodes as we only have a two-stage problem. We index all variables and equations (except the objective) by an extra set n which is recognized by GAMS/OSL-SE if it has an explanatory text of "*nodes*." All the first stage variables and equations are indexed by the element "*root*" and all second stage variables and equations are index by the element indicating the scenario. The complete model follows here:

Model stochoslse.gms. [3]

```
$ontext

   Transportation problem with stochastic demands.
   OSL-SE formulation

   Erwin Kalvelagen, January 2003
```

[3] http://www.amsterdamoptimization.com/models/twostage/stochoslse.gms

```
$offtext

sets
  i 'factories' /f1*f3/
  j 'distribution centers' /d1*d5/
  s 'individual scenarios' /lo,mid,hi/
;

parameter capacity(i) /f1 500, f2 450, f3 650/;

table demand(j,s) 'possible outcomes for demand'
      lo  mid   hi
  d1 150  160  170
  d2 100  120  135
  d3 250  270  300
  d4 300  325  350
  d5 600  700  800
;

table prob(j,s) 'probabilities for table demand'
      lo  mid   hi
  d1 .25  .50  .25
  d2 .25  .50  .25
  d3 .25  .50  .25
  d4 .30  .40  .30
  d5 .30  .40  .30
;

loop(j, abort$(abs(sum(s,prob(j,s))-1)>0.001) "probabilities don't add up");


*
* set up joint probabilities
* total scenarios =  3**5 = 243
*
set n 'nodes' /root,js1*js243/;
set js(n) /js1*js243/;
parameter jdemand(j,js) 'joint distribution: outcomes';
parameter jprob(js)   'joint distribution: probabilities';

alias (s,s1,s2,s3,s4,s5);
set current(js);
current('js1') = yes;
loop((s1,s2,s3,s4,s5),
   jdemand('d1',current) = demand('d1',s1);
   jdemand('d2',current) = demand('d2',s2);
   jdemand('d3',current) = demand('d3',s3);
   jdemand('d4',current) = demand('d4',s4);
   jdemand('d5',current) = demand('d5',s5);
   jprob(current) = prob('d1',s1)*prob('d2',s2)*prob('d3',s3)*prob('d4',s4)*prob('d5',s5);
   current(js) = current(js-1);
);
display jdemand,jprob;

scalar sumprob;
sumprob = sum(js, jprob(js));
display sumprob;
abort$(abs(sumprob-1)>0.00001) "joint probabilities don't add up";

table transcost(i,j) 'unit transportation cost'
       d1    d2    d3    d4    d5
  f1   2.49  5.21  3.76  4.85  2.07
  f2   1.46  2.54  1.83  1.86  4.76
  f3   3.26  3.08  2.60  3.76  4.45
;

scalar prodcost  'unit production cost' /14/;
scalar price     'sales price' /24/;
scalar wastecost 'cost of removal of overstocked products' /4/;
```

```
*---------------------------------------------------------------------
* We form here the deterministic equivalent
*---------------------------------------------------------------------

variables
   salesw(j,n)    'stochastic version of sales'
   wastew(j,n)    'stochastic version of waste'
   received(j,n)  'amount of product received in distribution center'
   ship(i,j,n)    'shipments'
   product(i,n)   'production'
   profit
;
positive variable ship,product,salesw,wastew;

equations
   objw
   sellingw(j,n)
   receive(j,n)
   production(i,n)
;

objw.. profit =e= sum((j,js),price*jprob(js)*salesw(j,js))
                - sum((i,j), transcost(i,j)*ship(i,j,'root'))
                - sum((j,js), wastecost*jprob(js)*wastew(j,js))
                - sum(i,prodcost*product(i,'root'));

receive(j,'root')..     received(j,'root') =e= sum(i, ship(i,j,'root'));
sellingw(j,js)..        received(j,'root') =e= salesw(j,js)+wastew(j,js);
production(i,'root')..  product(i,'root') =e= sum(j, ship(i,j,'root'));

product.up(i,'root') = capacity(i);
salesw.up(j,js) = jdemand(j,js);


option lp=oslse;
model stoch /objw,production,receive,sellingw/;
solve stoch maximizing profit using lp;

display ship.l,product.l,salesw.l,wastew.l;
```

Indeed this model is solved with an optimal objective of 10793.00.

6. Benders Decomposition

It is quite possible to develop a Benders Decomposition algorithm in GAMS [9]. Within the Benders framework, two different type of linear programming models need to be solved: a Master Problem that focuses on the first stage variables and a series of subproblems that deal with the second stage variables.

The Master Problem at iteration ν can be formulated as:

$$
\begin{aligned}
&\min c^T x + \theta \\
&Ax = b \\
&\theta \geq \sum_{\omega \in \Omega} p_\omega \left(-\overline{\pi}_\omega^\ell [T_\omega x + W_\omega \overline{y}_\omega^\ell - h_\omega]\right),\ \ell = 1, \ldots, \nu - 1 \\
&x \geq 0
\end{aligned} \tag{9}
$$

The constraint involving θ is the Benders' Optimality Cut. As all subproblems are always feasible, we do not need to include Feasibility Cuts in the Master Problem. The values $\overline{\pi}_\omega^\ell$ are the duals of the subproblem at iteration ℓ. Similarly, the quantities $\overline{y}_\omega^\ell$ are the optimal two-stage variables of the subproblems at iteration ℓ.

The subproblems at iteration ν take the form:

$$\begin{aligned} &\min d_\omega^T y_\omega \\ &W_\omega y_\omega = h_\omega - T_\omega \overline{x}^\nu \\ &y_\omega \geq 0 \end{aligned} \tag{10}$$

The values $\overline{x}^\nu$ are the values of the first stage variables as suggested by the Master Problem.

For the problem under consideration, a dual formulation of the subproblems is more interesting, as we can solve this problem without even invoking a linear programming solver. The primal subproblem can be formulated as:

$$\begin{aligned} \min\ &-\sum_j p_j Sales_j + \sum_j c_j Waste_j \\ &Sales_j + Waste_j = received_j \\ &Sales_j + SlackSales_j = demand_j \\ &Sales_j \geq 0, Waste_j \geq 0, SlackSales_j \geq 0 \end{aligned} \tag{11}$$

The dual of this problem is:

$$\begin{aligned} \max\ &\sum_j received_j \pi_j^{(1)} + \sum_j demand_j \pi_j^{(2)} \\ &\pi_j^{(1)} + \pi_j^{(2)} \leq -p_j \\ &\pi_j^{(1)} \leq c_j \\ &\pi_j^{(2)} \leq 0 \end{aligned} \tag{12}$$

which has the following optimal solution:

$$\pi_j^{(1)} = \begin{cases} c_j & \text{if } received_j > demand_j \\ -p_j & \text{otherwise} \end{cases} \tag{13}$$

and

$$\pi_j^{(2)} = \begin{cases} -p_j - c_j & \text{if } received_j > demand_j \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

The complete implementation of the Benders Algorithm is:

Model stochbenders.gms. [4]

```
sets
  i 'factories' /f1*f3/
  j 'distribution centers' /d1*d5/
  s 'individual scenarios' /lo,mid,hi/
;

parameter capacity(i) /f1 500, f2 450, f3 650/;

table demand(j,s) 'possible outcomes for demand'
      lo  mid   hi
  d1 150  160  170
  d2 100  120  135
  d3 250  270  300
  d4 300  325  350
  d5 600  700  800
;
```

[4] http://www.amsterdamoptimization.com/models/twostage/stochbenders.gms

```
table prob(j,s) 'probabilities for table demand'
      lo  mid   hi
  d1 .25  .50  .25
  d2 .25  .50  .25
  d3 .25  .50  .25
  d4 .30  .40  .30
  d5 .30  .40  .30
;

loop(j, abort$(abs(sum(s,prob(j,s))-1)>0.001) "probabilities don't add up");

*
* set up joint probabilities
* total scenarios =  3**5 = 243
*
set js 'scenarios for joint probabilities' /js1*js243/;
parameter jdemand(j,js) 'joint distribution: outcomes';
parameter jprob(js)   'joint distribution: probabilities';

alias (s,s1,s2,s3,s4,s5);
set current(js);
current('js1') = yes;
loop((s1,s2,s3,s4,s5),
   jdemand('d1',current) = demand('d1',s1);
   jdemand('d2',current) = demand('d2',s2);
   jdemand('d3',current) = demand('d3',s3);
   jdemand('d4',current) = demand('d4',s4);
   jdemand('d5',current) = demand('d5',s5);
   jprob(current) = prob('d1',s1)*prob('d2',s2)*prob('d3',s3)*prob('d4',s4)*prob('d5',s5);
   current(js) = current(js-1);
);
display jdemand,jprob;

table transcost(i,j) 'unit transportation cost'
       d1    d2    d3    d4    d5
  f1   2.49  5.21  3.76  4.85  2.07
  f2   1.46  2.54  1.83  1.86  4.76
  f3   3.26  3.08  2.60  3.76  4.45
;

scalar prodcost  'unit production cost' /14/;
scalar price     'sales price' /24/;
scalar wastecost 'cost of removal of overstocked products' /4/;


*---------------------------------------------------------------------
* Form the Benders master problem
*---------------------------------------------------------------------

set
   iter 'max Benders iterations' /iter1*iter25/
   dyniter(iter) 'dynamic subset'
;

positive variables
   ship(i,j)        'shipments'
   product(i)       'production'
   slackproduct(i)  'slack'
   received(j)      'quantity sent to market'
;
free variables
   zmaster          'objective variable of master problem'
   theta            'extra term in master obj'
;

equations
   masterobj        'master objective function'
   production(i)    'calculate production in each factory'
   receive(j)       'calculate quantity to be send to markets'
   prodcap(i)       'production capacity'
   optcut(iter)     'Benders optimality cuts'
```

```
;

parameter
   cutconst(iter)   'constants in optimality cuts'
   cutcoeff(iter,j) 'coefficients in optimality cuts'
;

masterobj..
    zmaster =e=  sum((i,j), transcost(i,j)*ship(i,j))
                + sum(i,prodcost*product(i)) + theta;

receive(j)..       received(j) =e= sum(i, ship(i,j));
production(i)..    product(i) =e= sum(j, ship(i,j));
prodcap(i)..       product(i) + slackproduct(i) =e= capacity(i);
optcut(dyniter)..  theta =g= cutconst(dyniter) +
                             sum(j, cutcoeff(dyniter,j)*received(j));

model masterproblem /masterobj, receive, production, prodcap, optcut/;

*---------------------------------------------------------------------
* Form the Benders' subproblem
* Notice in equation selling we use the level value received.l, i.e.
* this is a constant
*---------------------------------------------------------------------

positive variables
   sales(j)         'sales (actually sold)'
   waste(j)         'overstocked products'
   slacksales(j)    'slack'
;
free variables
   zsub             'objective variable of sub problem'
;

equations
   subobj           'subproblem objective function'
   selling(j)       'part of received is sold'
   selmax(j)        'upperbound on sales'
;

parameter demnd(j) 'demand';


subobj..
   zsub =e= -sum(j, price*sales(j)) + sum(j, wastecost*waste(j));

selling(j)..  sales(j) + waste(j) =e= received.l(j);
selmax(j)..   sales(j) + slacksales(j) =e= demnd(j);


model subproblem /subobj,selling,selmax/;

*---------------------------------------------------------------------
* Dual subproblem
*---------------------------------------------------------------------

variables
   pi_selling(j)    'dual of equation selling'
   pi_selmax(j)     'dual of equation selmax'
;

equations
   dualobj
   dualcon(j)
;

dualobj..
   zsub =e= sum(j, received.l(j)*pi_selling(j)) + sum(j, demnd(j)*pi_selmax(j));

dualcon(j).. pi_selling(j) + pi_selmax(j) =l= -price;

pi_selling.up(j) = wastecost;
```

```
pi_selmax.up(j) = 0;

model dualsubproblem /dualobj,dualcon/;



*---------------------------------------------------------------------
* Benders algorithm
*---------------------------------------------------------------------


*
* step 1: solve master without cuts
*
dyniter(iter) = NO;
cutconst(iter) = 0;
cutcoeff(iter,j) = 0;
theta.fx = 0;
solve masterproblem minimizing zmaster using lp;



*
* repair bounds
*
theta.lo = -INF;
theta.up = INF;

scalar lowerbound /-INF/;
scalar upperbound /INF/;
parameter objsub(js);
scalar objmaster;
objmaster = zmaster.l;


option limrow = 0;
option limcol = 0;
*subproblem.solprint = 0;
masterproblem.solprint = 0;

parameter pselling(j),pselmax(j);

loop(iter,

*
* solve subproblems
*
   dyniter(iter) = yes;
   loop(js,
       loop(j,
           if (received.l(j)>jdemand(j,js),
               pselling(j) = wastecost;
               pselmax(j) = -price-wastecost;
           else
               pselling(j) = -price;
               pselmax(j) = 0;
           );
       );
       objsub(js) = sum(j, pselling(j)*received.l(j)) + sum(j, pselmax(j)*jdemand(j,js));
       cutconst(iter) = cutconst(iter) - jprob(js)*sum(j,(-pselmax(j))*jdemand(j,js));
       cutcoeff(iter,j) = cutcoeff(iter,j) - jprob(js)*(-pselling(j));
   );

   upperbound = min(upperbound, objmaster + sum(js, jprob(js)*objsub(js)));

*
* convergence test
*
   display lowerbound,upperbound;
   abort$( (upperbound-lowerbound) < 0.0001*(1+abs(lowerbound)) ) "Converged";

*
```

```
* solve masterproblem
*
   solve masterproblem minimizing zmaster using lp;
   display ship.l;

   lowerbound = zmaster.l;
   objmaster = zmaster.l-theta.l;

);
```

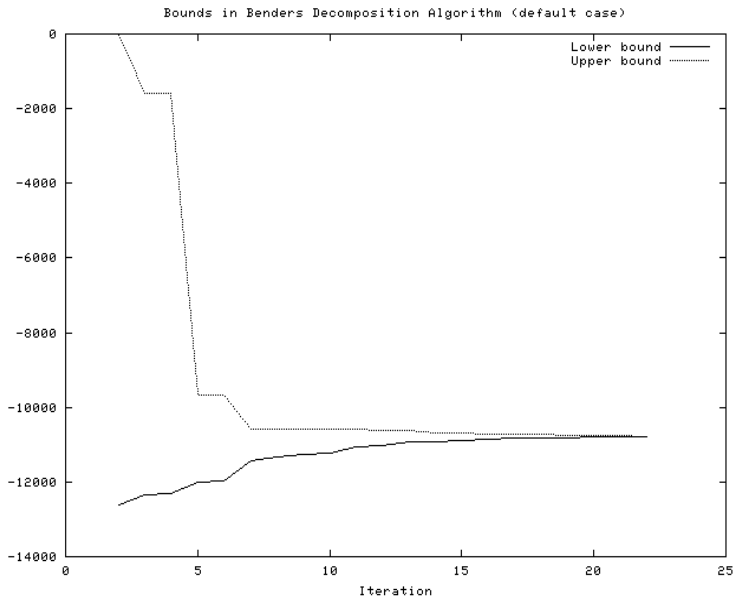

FIGURE 1. Benders lower and upper bound (default case)

The algorithm converges to a solution with an objective of 10793.00 in 22 iterations.

Notice that the primal and dual subproblems are formulated in this model. They are not actually used and are present for illustrative purposes only.

If we start the algorithm with the non-stochastic solution using the MID values, the problem solves somewhat faster in 18 iterations.

REFERENCES

1. J. R. Birge, M. A. H. Dempster, H. I. Gassmann, E. A. Gunn, A. J. King, and S. W. Wallace, *A standard input format for multiperiod stochastic linear programs*, Mathematical Programming Society COAL Newsletter (1987), no. 17, 1–19.
2. John R. Birge and François Louveaux, *Introduction to Stochastic Programming*, Springer Series in Operations Research, Springer, 1997.
3. GAMS Development Corp., *GAMS/OSL Stochastic Extensions User Notes*, `http://www.gams.com/docs/solver/oslse.pdf`, 2002.
4. H. I. Gassmann, *The SMPS format for stochastic linear programs*, `http://www.mgmt.dal.ca/sba/profs/hgassmann/SMPS2.htm`, 2002.
5. Gerd Infanger, *Planning Under Uncertainty – Solving Large-Scale Stochastic Linear Programs*, Boyd & Fraser, 1994.
6. _____, *DECIS User's Guide*, 1997.

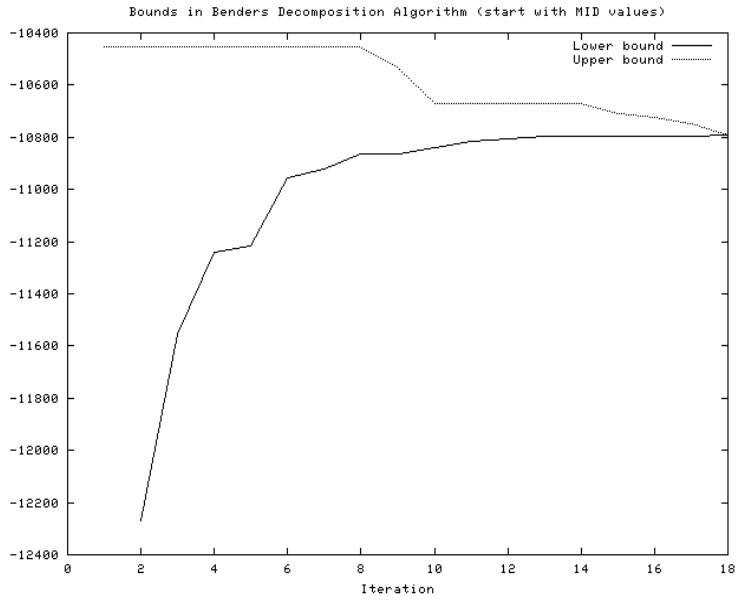

FIGURE 2. Benders lower and upper bound (start with MID values)

7. _____, *GAMS/DECIS User's Guide*, `http://www.gams.com/docs/solver/decis.pdf`, 1999.
8. Peter Kall and Stein W. Wallace, *Stochastic Programming*, Wiley, 1994.
9. Erwin Kalvelagen, *Benders Decomposition for Stochastic Programming with GAMS*, `http://www.amsterdamoptimization.com/pdf/stochbenders.pdf`, 2003.

AMSTERDAM OPTIMIZATION MODELING GROUP LLC, WASHINGTON DC/THE HAGUE
E-mail address: `erwin@amsterdamoptimization.com`