12

Decision making Under Uncertainty and Stochastic Programs

If you come to a fork in the road, take it.
-Y. Berra

12.1 Introduction

A big reason multiperiod planning is difficult is because of uncertainty about the future. For example, next year, if the demand for your new product proves to be large and the cost of raw material increases markedly, then buying a lot of raw material today would win you a lot of respect next year as a wise and perceptive manager. On the other hand, if the market disappears for both your product and your raw material, the company stockholders would probably not be so kind as to call your purchase of lots of raw material bad luck.

We apply the term stochastic program or scenario planning (SP) to any optimization problems (linear, nonlinear or mixed-integer) in which some of the model parameters are not known with certainty, and the uncertainty can be expressed with known probability distributions. Applications arise in a variety of industries:

- Financial portfolio planning over multiple periods for insurance and other financial companies, in face of uncertain prices, interest rates, and exchange rates,
- Exploration planning for petroleum companies,
- Fuel purchasing when facing uncertain future fuel demand,
- Fleet assignment: vehicle type to route assignment in face of uncertain route demand,
- Electricity generator unit commitment in face of uncertain demand,
- Hydro management and flood control in face of uncertain rainfall,
- Optimal time to exercise for options in face of uncertain prices,
- Capacity and Production planning in face of uncertain future demands and prices,
- Foundry metal blending in face of uncertain input scrap qualities,
- Product planning in face of future technology uncertainty,
- Revenue management in the hospitality and transport industries.

12.1.1 Identifying Sources of Uncertainty

In a discussion with the author, the chief financial officer of a large petroleum company made the comment: "The trouble with you academics is that you assume the probabilities add up to one. In the real world, they do not. You may think the possible outcomes are: 'hit oil', 'hit gas', or 'dry hole'. In reality, the drilling rig catches fire and causes a major disaster." The point of the comment is that an important part of managing uncertainty is identifying as many sources of uncertainty as possible. The following is a typical list with which to start:

1. Weather related:
   - decisions about how much and where to stockpile supplies of fuel and road salt in preparation for the winter;
   - water release decisions in the spring for a river and dam system, taking into account hydroelectric, navigation, and flooding considerations.
2. Financial uncertainty:
   - market price movements (e.g., stock price, interest rate, and foreign exchange rate movements);
   - defaults by business partner (e.g., bankruptcy of a major customer).
3. Political events:
   - changes of government;
   - outbreaks of hostilities.
4. Technology related:
   - whether a new technology is useable by the time the next version of a product is scheduled to be released.
5. Market related:
   - shifts or fads in tastes;
   - population shifts.
6. Competition:
   - incomplete knowledge of the kinds of strategies used by the competition next year.
7. Acts of God:
   - hurricane, tornado, earthquake, or fire;
   - equipment failure.

In an analysis of a decision under uncertainty, we would proceed through a list such as the above and identify those items that might interact with our decision. Weather, in particular, can be a big source of uncertainty. Hidroeléctrica Española, for example (see Dembo et al. (1990)), reports available power output per year from one of its hydroelectric facilities varied from 8,350 Gwh (Giga-watt-hours) to 2,100 Gwh over a three-year period simply because of rainfall variation.

Methods very similar to those described here have been used in the automotive industry to make plant opening and closing decisions in the face of uncertainty about future demand. These methods have also been used in the utilities industry to make fuel purchase decisions in the face of uncertainties about weather in the next few years.

12.2 The Scenario Planning (SP)Approach

We will start by considering planning problems with two periods. These situations consist of the following sequence of events:

1) We make a first-period decision.
2) Nature (frequently known as the marketplace) makes a random decision.
3) We make a second-period decision that attempts to repair the havoc wrought by nature in (2).

The scenario approach assumes there are a finite number of decisions nature can make. We call each of these possible states of nature a "scenario". For example, in practice, most people are satisfied with classifying demand for a product as being low, medium, or high; or classifying a winter as being severe, normal, or mild, rather than requiring a statement of the average daily temperature and total snowfall measured to six decimal places. General Motors has historically used low, medium, and high scenarios to represent demand uncertainty. The type of model we will describe for representing uncertainty in the context of LPs is called a "stochastic program". For a survey of applications and methodology, see Birge (1997). For an extensive introduction to stochastic programming ideas, see Kall and Wallace (1994). For a good discussion of some of the issues in applying stochastic programming to financial decisions, see Infanger (1994).

12.2.1 Formulation and Structure of an SP Problem

In decisionmaking under uncertainty, it is important to take into account the sequence in which information becomes available and we make decisions. We use the term *stage* to described the sequence pair: [1) new information becomes available, 2) we make a decision]. Usually, one can think of a stage as a 'time period', however there are situations where a stage may consist of several time periods. A stage: a) begins with one or more random events, e.g., some demands occur, and b) ends with our making one or more decisions, e.g., sell some excess product or order some more product.

Multistage decision making under uncertainty involves making optimal decisions for a T-stage horizon before uncertain events (random parameters) are revealed while trying to protect against unfavorable outcomes that could be observed in the future.

In its most general form, a multistage decision process with T+1 stages follows an alternating sequence of random events and decisions. Slightly more explicitly:

0.1) in stage 0, we make some initial decision, e.g., how much to order, taking into account that…

1.0) at the beginning of stage 1, "Nature" takes a set of random decisions, e.g., how much customers want to buy, leading to realizations of all random events in stage 1, and…

1.1) at the end of stage 1, having seen nature's decision, as well as our previous decision, we make a recourse decision, e.g., sell off excess product or order even more, taking into account that …

2.0) at the beginning of stage 2, "Nature" takes a set of random decisions, leading to realizations of all random events in stage-2, and…

2.1) at the end of stage 2, having seen nature's decision, as well as our previous decisions, we make another recourse decision taking into account that …

.
.
.

T.0) At the beginning of stage *T*, "Nature" takes a random decision, leading to realizations of all random events in stage *T*, and…

T.1) at the end of stage *T*, having seen all of nature's *T* previous decisions, as well as all our previous decisions, we make the final recourse decision.

The decision taken in stage 0 is called the *initial decision*, whereas decisions taken in succeeding stages are sometimes called *recourse decisions*. Recourse decisions are interpreted as corrective actions that are based on the actual values the random parameters realized so far, as well as the past decisions taken thus far.

The essential steps in formulating an SP in LINGO are:

1) Write a standard deterministic model (the core model) as if the random variables are variables or parameters.

2) Identify the random variables, and decision variables,and their staging. This is done using a statement like **`@SPSTGVAR( 0, Q)`** to declare that *Q* is a stage 0 decision variable, and a statement like **`@SPSTGRNDV( 1, DEMAND)`** to declare that *DEMAND* is a random variable in stage 1.

3) Provide the distributions describing the random variables. Distribution specification is specific using a function of the form **`@SPDIST*( parameters, randomvariable)`**. For example, **`@SPDISTPOIS( 60, DEMAND)`** means that *DEMAND* is a random variable from a Poisson distribution with mean 60.

4) Specify manner of sampling from the distributions, (mainly the sample size). This information is provided via a statement like: **`@SPSAMPSIZE( 1, 200),`** meaning that in stage 1 a sample size of 200 should be used.

5) List the variables for which we want a scenario by scenario report or a histogram: WBSP_REP(cell_list) for scenario list of values, or

WBSP_HIST(bins, cell) for histograms.

12.3 Single Stage Decisions Under Uncertainty

The simplest problems of decision making under uncertainty involve the case where there is but a single stage with randomness.

12.3.1 The News Vendor Problem

The simplest problem of decision making under uncertainty is the News Vendor problem, i.e., we must decide how much to stock in anticipation of demand, before knowing exactly what the demand will be. Below we see how the Newsvendor problem is set up as a LINGO model.

```
MODEL:                  !(SP_NBsimpleL.lg4);
 ! Newsvendor problem as a stochastic program in LINGO.
   How much should we stock, Q,
   in anticipation of uncertain DEMAND?
  Parameters:
    10 = cost/unit purchased and stocked,
    15 = revenue/unit sold;

! Step 1: Core model definition--------------------------------+;
   [R_OBJ] MAX = PROFIT;
   ! (Expected) Profit =
       sales - purchase cost;
   PROFIT = 15 * SALES - 10 * Q;
   @FREE( PROFIT);  ! Allow negative PROFIT because
                    might have a loss in some scenarios;

   ! Set excess inventory or shortage ;
   EXCESS - SHORT = Q - DEMAND;
   SALES = DEMAND - SHORT;

! SP related declarations -------------------------------------+;
! Step 2: staging info;
   @SPSTGVAR( 0, Q); ! Amount to purchase, Q, is a stage 0 decision;
   @SPSTGRNDV( 1, DEMAND); ! Demand is a random variable observed
                              in stage 1, at the beginning;

! Step 3: Distribution info;
   @SPDISTPOIS( 60, DEMAND); ! Demand has a Poisson distribution;
  ! @SPDISTNORM( 60, 12, DEMAND); ! Demand has Normal distribution;

! Step 4: Sampling structure;
   @SPSAMPSIZE( 1, 200);  ! Specify the stage 1 sample size;
END
```

Part of the solution report is shown below.

```
  Global optimal solution found.
```

```
Objective value:                                257.8500
Infeasibilities:                                0.000000
Total solver iterations:                             134
Elapsed runtime seconds:                            0.49

Expected value of:
   Objective (EV):                               257.8500
   Wait-and-see model's objective (WS):          299.8000
   Perfect information (EVPI = |EV - WS|):       41.95000
   Policy based on mean outcome (EM):            253.5880
   Modeling uncertainty (EVMU = |EM - EV|):      4.262000


Stage 0 Solution
----------------
        Variable        Value
               Q     57.00000

Staging Report
--------------
 Random Variable        Stage
          DEMAND        1

        Variable        Stage
          PROFIT   1*
           SALES   1*
               Q   0
          EXCESS   1*
           SHORT   1*

    (*) Stage was inferred

Random Variable Distribution Report
-----------------------------------
                                  Sample        Sample
Random Variable          Mean     StdDev      Distribution
         DEMAND      59.96000    7.740052      POISSON,60

Scenario: 1   Probability: 0.5000000E-02   Objective: -45.00000
---------------------------------------------------------------
Random Variable         Value
         DEMAND      35.00000

       Variable         Value
         PROFIT     -45.00000
          SALES      35.00000
              Q      57.00000
          EXCESS     22.00000
           SHORT     0.000000


Scenario: 2   Probability: 0.5000000E-02   Objective: 285.0000
---------------------------------------------------------------
```

```
  Random Variable          Value
            DEMAND       66.00000

         Variable          Value
           PROFIT       285.0000
            SALES       57.00000
                Q       57.00000
           EXCESS       0.000000
            SHORT       9.000000
                  .   .   .
```

12.3.2 Multi-product Inventory with Repositioning

This is example is a very simplified illustration of an inventory management approach used by some apparel retailers. The general sequence of events is:

Stage 0) Before the selling season starts
the retailer commits inventory to a number of locations and/or products.
Stage 1, beginning) Demands at the various locations or products are observed.
Stage 1, end) Product can be repositioned to some extent, at some additional cost, among the various locations/products, generally moving inventory to the locations/products with higher than expected demand.

This is a very crude simplified representation of an inventory allocation system with reallocation used at the clothing retailers Sport Obermeyer, see Fisher and Raman(1996) and at the Spanish firm Zara, see Caro and Gallien (2010). Our example below is closer to that of Sport Obermeyer, where the secondary reallocation is over products, whereas at Zara, the reallocation is over locations. In the example below, in stage 0 we need to decide what initial quantities should be produced to inventory of three types of parkas, the "Anita", "Daphne", and "Electra". After this initial production run, we observe the demands for the three parkas. Once we see the demands, we have access to a fast backup production facility of limited capacity that can produce any of the three products. Although this backup facility is fast, it is also very expensive per unit produced, and it has limited capacity, so if we had perfectly accurate forecasts, we would not use the backup facility. We would produce just the right amount of each product from the outset. In the real world where perfect forecasts are the exception, the main question is: How much should we produce of each product initially, taking into account that we can use the somewhat expensive backup facility to partially compensate for our forecast errors.

In the previous example, we sampled from a standard distribution, e.g., the Poisson. In this example, we illustrate using a table of demand scenarios. There are four possible scenarios with associated probabilities.

```
! Capacity Planning with Re-positioning Under Uncertainty (SP_Cap_Plan_Gen);
!  Stage 0: We decide what inventories or capacities
            to place at various origins.
   Stage 1 beginning: Demands at various demand points observed.
   Stage 1       end: We satisfy demands from available quantities
                      (by solving a transportation problem.);
SETS:
 ORIGIN: CPERU, ULCAP, Q;
 DESTN: DEMAND;
 OXD( ORIGIN, DESTN): PROFC, X;
 SCENE: PROB;
```

```
 SXD( SCENE, DESTN): RDEM;
ENDSETS
DATA:
! These data are based very loosely on the Sport Obermeyer apparel
problem studied by Fisher and Raman, Operations Research, vol. 44, no. 1;
 ORIGIN = OANITA, ODAPHNE, OELECTRA, OGENRIC; ! Sources of production;
  CPERU =     80       90        65        5; ! Cost per unit to commit;
  ULCAP =   9999     9999      9999      150; ! Upper limit on production;
 DESTN = DANITA, DDAPHNE, DELECTRA; ! Demand points;
 PROFC = 180      0        0     ! Incremental profit from satisfying a ;
           0    160        0     ! particular demand point from a ;
           0      0      140     ! particular supply point;
          90     50,      60;    ! Generic is a quick response source;
 ! E.g., OANITA can satisfy only DANITA, ODAPHNE only DDAPHNE, etc.,
         OGENERIC can satisfy any demand but is not as profitable and
         has limited capacity;
 PROB = 0.2 0.3 0.4 0.1;  ! Probabilities of various scenarios;
 RDEM = 300 400 400       ! The demand scenarios;
        320 370 433
        333 383 460
        500 320 610;
ENDDATA
! Decision variables:
   Q(i) = amount of initial inventory or capacity we put at or in
          source i before seeing demand.
   X(i,j) = amount we reposition from source i to destinantion j
          after seeing demand. Some destinations may also be sources;

! 1) Define core model;
  !   Maximize revenue from sales minus cost of producing inventory;
    MAX = @SUM( OXD(i,j): PROFC(i,j)*X(i,j))
        - @SUM( ORIGIN(i): CPERU(i)* Q(i));

  ! Cannot install/produce more than upper limit;
   @FOR( ORIGIN(i):
      Q(i) <= ULCAP(i);
       );
  ! Cannot sell more than we stock;
   @FOR( ORIGIN(i):
     @SUM( DESTN(j): X(i,j)) <= Q(i);
       );
  ! Cannot sell more than demand;
   @FOR( DESTN(j):
     @SUM( ORIGIN(i): X(i,j)) <= DEMAND(j);
       );

 ! 2) Specify staging of decisions. Click on
        Edit | Paste function | Stochastic Programming
      to see choices and syntax;
   @FOR( ORIGIN(i):
      @SPSTGVAR( 0, Q(i)); ! The Q's are a stage 0 decision;
       );
   ! ... and demands (stage 1);
```

```
 @FOR( DESTN(j):
    @SPSTGRNDV( 1, DEMAND(j));  ! Demands observed in stage 1 beginning;
     );

! 3) Specify distribution of demands.
    Demand scenarios come from a table;
    @SPDISTTABLE( RDEM, DEMAND, PROB);

! 4) Number of scenarios;
     @SPSAMPSIZE( 1, 10);
```

A portion of the solution report appears below. Notice that all of the available generic capacity is committed.

```
Global optimal solution found.
Objective value:                              90207.00

Stage 0 Solution
----------------

            Variable           Value
          Q( OANITA)        320.0000
         Q( ODAPHNE)        370.0000
        Q( OELECTRA)        433.0000
         Q( OGENRIC)        150.0000

Scenario: 1   Probability: 0.1000000   Objective: 82905.00
-------------------------------------------------------------

     Random Variable           Value
     DEMAND( DANITA)        300.0000
    DEMAND( DDAPHNE)        400.0000
   DEMAND( DELECTRA)        400.0000

            Variable           Value
X( OGENRIC, DDAPHNE)        30.00000

Scenario: 2   Probability: 0.1000000   Objective: 93065.00
-------------------------------------------------------------

       Random Variable           Value
       DEMAND( DANITA)        333.0000
      DEMAND( DDAPHNE)        383.0000
     DEMAND( DELECTRA)        460.0000

              Variable           Value
   X( OGENRIC, DANITA)        13.00000
  X( OGENRIC, DDAPHNE)        13.00000
 X( OGENRIC, DELECTRA)        27.00000

Scenario: 3   Probability: 0.1000000   Objective: 89625.00
-------------------------------------------------------------

       Random Variable           Value
```

```
        DEMAND( DANITA)        320.0000
       DEMAND( DDAPHNE)        370.0000
      DEMAND( DELECTRA)        433.0000

                Variable           Value


Scenario: 4   Probability: 0.1000000   Objective: 93065.00
-------------------------------------------------------------

       Random Variable           Value
        DEMAND( DANITA)        333.0000
       DEMAND( DDAPHNE)        383.0000
      DEMAND( DELECTRA)        460.0000

                Variable           Value
    X( OGENRIC, DANITA)        13.00000
   X( OGENRIC, DDAPHNE)        13.00000
  X( OGENRIC, DELECTRA)        27.00000


Scenario: 8   Probability: 0.1000000   Objective: 95125.00
-------------------------------------------------------------

       Random Variable           Value
        DEMAND( DANITA)        500.0000
       DEMAND( DDAPHNE)        320.0000
      DEMAND( DELECTRA)        610.0000

                Variable           Value
    X( OGENRIC, DANITA)        150.0000
```

12.4 Multi-Stage Decisions Under Uncertainty

Our examples thus far have been at most two stages. In stage 0, we make a decision, and then in stage 1 at the beginning there is one occurrence of a random event, and then finally we make one recourse decision. A slightly more complicated class of problems is the set of problems in which there are two or more separate random stages, with an intervening set of decisions. Perhaps the simplest multi-stage problems of decision making under risk are “stopping “ problems, examined next.

12.4.1 Stopping Rule and Option to Exercise Problems

Some sequential decision problems are of the form: a) Each period we have to make an accept or reject decision; b) once we accept, the "game is over". We then have to live with that decision. Our next example is the simplest example of a problem known variously as a stopping problem, the college acceptance problem, the secretary problem, or the dating game. The general situation is as follows. Each period we are offered an object of known quality. We have a choice of either a) accept the object and end the game, or b) reject the object and continue in the hope that a better object will become available in a future period. The problem is nontrivial because we do not know the qualities in advance. The following illustrates. Each period we will see either a 2, a 7, or a 10, where 10 is the best possible, and 2 is the worst. It is clear that once we see a 10, we might as well accept. We can never do better. If we see a 2, we should never accept unless it is the last period. Whether we should accept or reject a 7 in intermediate periods is at the moment a puzzle, depending upon the probabilities of the various outcomes. There are four periods, i.e., we have 4 chances. The completely deterministic "core" model is quite simple, namely:

Maximize $v_1 * y_1 + v_2 * y_2 + v_3 * y_3 + v_4 * y_4$;

subject to:

$$y_1 + y_2 + y_3 + y_4 \leq 1;$$

$$y_j = 0 \text{ or } 1, \text{ for } j = 1, 2, 3, 4;$$

The complication is that we do not know the v_j in advance. In particular, we must choose the value for y_j immediately after seeing v_j, without knowing the future v_j 's. If we follow the simple rule of accepting the first candidate, i.e., setting = 1, then the expected value of the objective function is (2+7+10)/3 = 6.3333. To check our understanding, we might ask ourselves several questions. How much better than 6.3333 can we do by being more thoughtful? What will the optimal policy look like? We can deduce certain features of it, such as: 1) If we see a 10, then accept it immediately. We can do no better; 2) If we see a 2, reject it, except if it is the last period, then accept. The big question is what to do when we see a 7 in any period before the last. The model formulated in LINGO appears below.

```
! The  Dating Game                                    (SP_DatinGame)

  We interview one prospect per stage or period.
 The quality of a prospect is a random variable.

 After interviewing a prospect we must make an Accept or Reject decision.

 Once we Accept, the game is over.
 We want to maximize the expected quality of the accepted prospect;

SETS:
  PERIOD: QUALITY, Y, SAMP_SZ;
  QPOSS: QOFP;
ENDSETS

DATA:
  PERIOD =  P1 P2 P3 P4; ! The view and accept/reject periods);
  SAMP_SZ=   3  3  3  3; ! Sample size for each period;
  QOFP = 2  7  10;  ! Possible qualities of prospects;
```

```
ENDDATA

! 1) Core Model ---------------------------------------------+;
 ! Y(p) = 1 if we accept the prospect we see in stage p;
! Maximize the quality of the accepted prospect;
   MAX = @SUM( PERIOD(p): QUALITY(p)*Y(p));

! We can accept only once;
   @SUM( PERIOD( P): Y( P)) <= 1;

! We either accept or reject, no halfsies;
   @FOR( PERIOD(p): @BIN( Y(p)) );

!LS*** These redundant constraints are added just to make the solver happy;
  @SPSTGVAR( 0, Y0); ! In case LINGO wants a decision variable in stage 0;
   Y0 + Y(1) <= 1;
   Y0 + Y(2) <= 1;
   Y0 + Y(3) <= 1;

! SP Related Declarations ----------------------------+;
!  2) Staging information;
!   QUALITY is a random variable;
@FOR( PERIOD( p):
   @SPSTGRNDV( p, QUALITY( p));
! The decision variables;
   @SPSTGVAR( p, Y( p));
    );

! 3) Declare a table distribution;
@FOR( PERIOD( P) :
   ! Quality is chosen randomly from table QOFP;
   @SPDISTTABLE( QOFP, QUALITY( p));
    );

 ! 4) Declare sample size for each stage/period;
@FOR( PERIOD( P):
   ! Set the sample size for the stage;
   @SPSAMPSIZE( P, SAMP_SZ( P));
    );
```

When solved, we see that the expected objective value is 9.012346, quite a bit better than the 5.333333 we would get by taking the first offer.

With regard to the policy, in particular, what to do when we are offered a “7”, we can look Scenario 51 below.

```
  Global optimal solution found.
  Objective value:                              9.012346

  Scenario: 61   Probability: 0.1234568E-01   Objective: 7.000000
```

```
------------------------------------------------------------------

       Random Variable           Value
          QUALITY( P1)         7.000000
          QUALITY( P2)         7.000000
          QUALITY( P3)         7.000000
          QUALITY( P4)         10.00000

              Variable           Value
              Y( P3)         1.000000
```

Notice from the highlighted row, for the given probabilities, if we see a 7 in stage 1 or 2, we do not accept (0) it, however, when we see a 7 in stage 3, whe accept (1).

12.4.2. An Option Exercise Stopping Problem

In financial markets it is frequently possible to buy options to buy or sell some financial instrument at an agreed upon "strike" price. This is a type of stopping problem. Once we have exercised the option, the game is over. The option exercise problem differs from our previous stopping problem example only in the manner in which the random variables, in this case the price of the financial instrument, is determined. In this particular example we will have five periods/stages/decision points, so the core model is similar to before:

Maximize $v_1y_1 + v_2y_2 + v_3y_3 + v_3y_3 + v_5y_5$;

subject to:

$y_1 + y_2 + y_3 + y_4 + y_5 \leq 1$;

$y_j = 0$ or 1, for $j = 1, 2, 3, 4, 5$;

The difference is the manner in which the v_j are determined. In this particular example, we assume that with equal probability the financial instrument, say a stock, changes each period by either 1) increasing by 6%, or 2) increases, by 1%, or 3) decreases by 4%. Further, we have to pay for the option up front, however, if and when we exercise the option, we get paid (difference between the strike price minus the then current price) only later at the point of exercise. Therefore, we want to discount the future cash inflow back to the point in time that we purchase the option. Figure 5.8 shows the setup in What's*Best*!.

When solved, from the WB! Status tab, we see that the expected value of the objective is 1.669324. This means, that we would be we be willing to pay up to about 1.67 for this option. One of the attractive features of using stochastic programming is that you get to see the distribution of the profit. If we look on the WB!_Histogram tab, we see the histogram in Figure 5.9. The interesting message from this histogram is that even though the expected profit contribution from exercising the option is about 1.67, we should expect a profit contribution of zero about 70% of the time.

With regard the policy of when to sell, recall that the strike price was 99, so we would never sell if the price > 99. From looking at the WB_Stochastic tab in Figure 5.10, we see that the policy is:

Stage	Sell at Strike Price if Market Price ≤
1	never
2	92.16
3	93.08
4	94.01
5	99.

12.5 Expected Value of Perfect Information (EVPI)

Uncertainty has its costs. Therefore, it may be worthwhile to invest in research to reduce the uncertainty. This investment might be in better weather forecasts for problems like the one just considered or it might be in test markets or market surveys for problems relating to new product investment. A bound on the value of better forecasts is obtainable by considering the possibility of getting perfect forecasts, so-called perfect information.

We have sufficient information on the snow removal problem to calculate the value of perfect information. For example, if we knew beforehand the winter would be warm, then we saw from the solution of the warm winter model the total cost would be \$583,333.3. On the other hand, if we knew beforehand that the winter would be cold, we saw the total cost would be \$970,000. Having perfect forecasts will not change the frequency of warm and cold winters. They will presumably still occur with respective probabilities 0.4 and 0.6. Thus, if we had perfect forecasts, the expected cost per season would be:

$$0.4 \times 583{,}333.3 + 0.6 \times 970{,}000 = 815{,}333.3$$

From the solution of the complete model, we see the expected cost per season without any additional information is \$819,888.3. Thus, the expected value of perfect information is 819,888.3 – 815,333.3 = \$4,555.0. Therefore, if a well-dressed weather forecaster claims prior knowledge of the severity of the coming winter, then an offer of at most \$4,555 should be made to learn his forecast. We say the expected value of perfect information in this case is \$4,555. In reality, his forecast is probably worth considerably less because it is probably not a perfect forecast.

12.6 Expected Value of Modeling Uncertainty

Suppose the EVPI is high. Does this mean it is important to use stochastic programming, or the scenario approach? Definitely not. Even though the EVPI may be high, it may be a very simple deterministic model does just as well (e.g., recommends the same decision as a sophisticated stochastic model). The Expected Value of Modeling Uncertainty (EVMU) measures the additional profit possible by using a "correct" stochastic model. EVMU is always measured relative to some simpler deterministic model.

12.6.1 Certainty Equivalence

An interesting question is: are there some situations in which we know in advance EVMU = 0?. A roughly equivalent question is: "Under what conditions can we replace a random variable in a model by its expected value without changing the action recommended by the model?" If we can justify such a replacement, then we have *a priori* determined that the EVMU is zero for that random variable. The following gives a sufficient condition:

> *Certainty Equivalence Theorem*: If the randomness or unpredictability in problem data exists solely in the objective function coefficients of a linear objective function, then

it is correct to solve the model in regular form after simply using the expected values for the random coefficients in the objective.

If the randomness exists in a right-hand side or a constraint coefficient, then it is generally not correct to simply replace the random element by its average or expected value. We can be slightly more precise if we define:

X = the set of decision variables,
Y_i = some random variable in the model,

$\bar{Y}_i$ = all other random variables in the model, except Y_i,

$\tilde{Y}_i$ = all other random variables that are independent of Y_i.

We are justified in replacing Y_i by its expected value, $E(Y_i)$, if Y_i appears only in the objective function, and each term containing Y_i either:

is not a function of X, or
is linear in Y_i and contains no random variables dependent upon Y_i.

Equivalently, we must be able to write the objective as:

$$\text{Min } F_1(X, \bar{Y}_i) + F_2(X, \tilde{Y}_i) * Y_i + F_3(\bar{Y}_i, Y_i).$$

If we take expected values:

$$E[F_1(X,Y_i) + F_2(X, \tilde{Y}_i) * Y_i + F_3(\bar{Y}_i, Y_i)]$$
$$= E[F_1(X, \bar{Y}_i)] + E[F_2(X, \tilde{Y}_1)] * E(Y_i) + E[F_3(\bar{Y}_i, Y_i)].$$

The third term is a constant with respect to the decision variables, so it can be dropped.

Thus, any X that minimizes $E[F_1(X \bar{Y}_i) + F_2(X, \tilde{Y}_i) * Y_i]$ also minimizes:

$$E[F_1(X, \bar{Y}_i) + F_2(X, \tilde{Y}_i) * E(Y_i)].$$

As an example, consider a farmer who must decide how much corn, beans, and milo to plant in the face of random yields and random prices for the crops. Further, suppose the farmer receives a government subsidy that is a complicated function of current crop prices and the farmer's total land holdings, but not a function of current yield or planting decisions. Suppose the price for corn at harvest time is independent of the yield. The farmer's income can be written (income from beans and milo) + (acreage devoted to corn) × (corn yield) × (price of corn) + (subsidy based on prices).

The third term is independent of this year's decision, so it can be disregarded. In the middle term, the random variable, "price of corn", can be replaced by its expected value because it is independent of the two other components of the middle term.

12.7 Risk Aversion

Thus far, we have assumed the decision maker is strictly an expected profit maximizer and is neither risk averse nor risk preferring. Casino gamblers who play games such as roulette must be risk preferring

if the roulette wheel is not defective, because their expected profits are negative. A person is risk averse if he or she attaches more weight to a large loss than expected profits maximization would dictate.

In the context of the snow removal problem, the Streets Manager might be embarrassed by a high cost of snow removal in a cold winter even though long run expected cost minimization would imply an occasional big loss. From the optimal policy for the snow removal problem, we can see the sum of first-period plus second-period costs if the winter is cold is:

```
70*BF1 + 20*BS1 + KC = 977591.
```

On the other hand, if it is known beforehand the winter will be cold, then we have seen this cost can be reduced to $970,000.

For fear of attracting the attention of a political opponent, the Streets Manager might wish to prevent the possibility of a cost more than $5,000 greater than the minimum possible for the cold winter outcome.

The Manager can incorporate his risk aversion into the LP by adding the constraint:

```
70 * BF1 + 20 * BS1 + KC ≤ 975000.
```

When this is done, the solution is:

```
Optimal solution found at step:        11
Objective value:                 820061.1
Variable           Value        Reduced Cost
     BF1        3780.556           0.0000000
     BS1        2916.667           0.0000000
      KW        264680.6           0.0000000
      KC        652027.8           0.0000000
     BFW       0.0000000            3.200000
     XFW        863.8889           0.0000000
      PW       0.0000000            4.611115
      SW        2916.667           0.0000000
     BSW       0.0000000            3.533334
     XSW       0.0000000            8.466666
     BFC        1027.778           0.0000000
     XFC       0.0000000            5.333333
      PC        1891.667           0.0000000
      SC        2916.667           0.0000000
     BSC       0.0000000            8.466667
     XSC       0.0000000            2.866666
```

The expected cost has increased by about 820,061 – 819,888 = 173 dollars. A politician might consider this a price worth paying to reduce his worst case (cold winter) cost almost $2,600. Notice, however, performance in the event of a warm winter does not look as good. The value of *XFW* indicates there will be almost 864 units of fuel to be disposed of at the end of a warm winter.

12.7.1 Downside Risk

There is a variety of ways of measuring risk. Variance is probably the most common measure of risk. The variance measure gives equal weight to deviations above the mean as well as below. For a symmetric distribution, this is fine, but for nonsymmetrical distributions, this is not attractive. Most people worry a lot more about returns that are below average than ones above average.

Downside risk is a reasonably intuitive way of quantifying risk that looks only at returns lower than some threshold. In words, downside risk is the expected amount by which return falls short of a specified target. To explain it more carefully, define:

P_s = probability that scenario s occurs
T = a target return threshold that we specify
R_s = the return achieved if scenario s occurs
D_s = the down side if scenario s occurs
= max $\{0, T - R_s\}$
ER = expected downside risk
= $p_1D_1 + p_2D_2 + ...$

12.7.2 Example

Suppose the farmer of our earlier acquaintance has made two changes in his assessment of things: (a) he assesses the probability of a wet season as 0.7 and (b) he has eliminated beans as a possible crop, so he has only two choices (corn and sorghum). A reformulation of his model is:

```
MAX = 0.7 * RW + 0.3 * RD;
RW - 100 * C - 70 * S = 0;
RD + 10 * C - 40 * S = 0;
C + S = 1;
@FREE(RW);
@FREE(RD);
```

The variables *RW* and *RD* are the return (i.e., profit) if the season is wet or dry, respectively. Notice both *RW* and *RD* were declared as FREE, because *RD* in particular could be negative.

When solved, the recommendation is to plant 100% corn with a resulting expected profit of 67:

```
Optimal solution found at step:          0
Objective value:                  67.00000

Variable           Value        Reduced Cost
      RW        100.0000           0.0000000
      RD       -10.00000           0.0000000
       C        1.000000           0.0000000
       S       0.0000000            6.000000

     Row    Slack or Surplus      Dual Price
       1         67.00000           1.000000
       2       0.0000000           0.7000000
       3       0.0000000           0.3000000
       4       0.0000000            67.00000
```

The solution makes it explicit that, if the season is dry, our profits (*RD*) will be negative. Let us compute the expected downside risk for a solution to this problem. We must choose a target threshold. A plausible value for this target is one such that the most conservative decision available to us just barely has an expected downside risk of zero. For our farmer, the most conservative decision is sorghum. A target value of 40 would give sorghum a downside risk of zero. To compute the expected downside risk for our problem, we want to add the following constraints:

```
DW ≥ 40 – RW
DD ≥ 40 – RD
ER = .7 DW + .3 DD
```

The constraint $DW \geq 40 - RW$ effectively sets $DW = \max(0, 40 - RW)$.

When converted to standard form and appended to our model, we get:

```
MAX = 0.7 * RW + 0.3 * RD;
RW - 100 * C - 70 * S = 0;
RD + 10 * C - 40 * S = 0;
C + S = 1;
RW + DW > 40;
RD + DD > 40;
- 0.7 * DW - 0.3 * DD + ER = 0;
@FREE(ER);
@FREE(RW);
@FREE(RD);
```

The solution is:

```
Optimal solution found at step:         2
Objective value:                 67.00000

Variable           Value        Reduced Cost
      RW        100.0000           0.0000000
      RD       -10.00000           0.0000000
       C        1.000000           0.0000000
       S       0.0000000            6.000000
      DW       0.0000000           0.0000000
      DD        50.00000           0.0000000
      ER        15.00000           0.0000000

     Row    Slack or Surplus      Dual Price
       1        67.00000            1.000000
       2       0.0000000           0.7000000
       3       0.0000000           0.3000000
       4       0.0000000            67.00000
       5        60.00000           0.0000000
       6       0.0000000           0.0000000
       7       0.0000000           0.0000000
```

Because we put no constraint on expected downside risk, we get the same solution as before, but with the additional information that the expected downside risk is 15.

What happens as we become more risk averse? Suppose we add the constraint $ER \leq 10$. We then get the solution:

```
Optimal solution found at step:         2
Objective value:                 65.00000

Variable           Value        Reduced Cost
      RW        90.00000           0.0000000
      RD        6.666667           0.0000000
       C       0.6666667           0.0000000
       S       0.3333333           0.0000000
      DW       0.0000000           0.2800000
      DD        33.33333           0.0000000
      ER        10.00000           0.0000000
```

Notice the recommendation is now to put 1/3 of the land into sorghum. The profit drops modestly to 65 from 67. If the season is dry, the profit is now 6.67 rather than −10 as before. Finally, let's constrain the expected downside risk to zero with $ER \leq 0$. Then the solution is:

```
Optimal solution found at step:          2
Objective value:                  61.00000

Variable           Value        Reduced Cost
      RW        70.00000           0.0000000
      RD        40.00000           0.0000000
       C       0.0000000           0.0000000
       S        1.000000           0.0000000
      DW       0.0000000           0.2800000
      DD       0.0000000           0.0000000
      ER       0.0000000           0.0000000

     Row    Slack or Surplus      Dual Price
       1        61.00000            1.000000
       2       0.0000000           0.7000000
       3       0.0000000           0.4200000
       4       0.0000000            65.80000
       5        30.00000           0.0000000
       6       0.0000000          -0.1200000
       7       0.0000000          -0.4000000
       8       0.0000000           0.4000000
```

Now, all the land is planted with sorghum and expected profit drops to 61.

12.8 Dynamic Programming and Financial Option Models

The term *dynamic programming* is frequently applied to the solution method described above. We illustrate it with three examples in the area of financial options. One in the stock market, one in the bond market, and the third in foreign exchange. A stock option is a right to buy a specified stock at a specified price (the so-called strike price) either on a specified date (a so-called European option) or over a specified interval of time (a so-called American option). An interesting problem in finance is the determination of the proper price for such an option. This problem was "solved" by Black and Scholes (1973). Below is a LINGO implementation of the "binomial pricing" version of the Black/Scholes model:

```
MODEL:
SETS:                                !(OPTONB);
! Binomial option pricing model: We assume that
a stock can either go up in value from one period
to the next with probability PUP, or down with
probability (1 - PUP). Under this assumption,
a stock's return will be binomially distributed.
  We can then build a
dynamic programming recursion to
determine the option's value;
! No. of periods, e.g., weeks;
   PERIOD /1..20/:;
ENDSETS
DATA:
! Current price of the stock;
   PNOW   = 40.75;
! Exercise price at option expiration;
   STRIKE = 40;
! Yearly interest rate;
   IRATE  = .163;
! Weekly variance in log of price;
   WVAR   = .005216191 ;
ENDDATA
SETS:
!Generate our state matrix for the DP.STATE(S,T) may
be entered from STATE(S,T-1)if the stock lost value,
or it may be entered from STATE(S-1,T-1) if stock
 gained;
   STATE( PERIOD, PERIOD)| &1 #LE# &2:
      PRICE,    ! There is a stock price, and   ;
      VAL;      ! a value of the option;
ENDSETS
! Compute number of periods;
   LASTP = @SIZE( PERIOD);
! Get the weekly interest rate;
   ( 1 + WRATE) ^ 52 = ( 1 + IRATE);
! The weekly discount factor;
   DISF = 1/( 1 + WRATE);
! Use the fact that if LOG( P) is normal with
  mean LOGM and variance WVAR, then P has
  mean EXP( LOGM + WVAR/2), solving for LOGM...;
   LOGM = @LOG( 1 + WRATE) - WVAR/ 2;
```

```
! Get the log of the up factor;
   LUPF = ( LOGM * LOGM + WVAR) ^ .5;
! The actual up move factor;
   UPF = @EXP( LUPF);
! and the down move factor;
   DNF = 1/ UPF;
! Probability of an up move;
   PUP =  .5 * ( 1 + LOGM/ LUPF);
! Initialize the price table;
   PRICE( 1, 1) = PNOW;
! First the states where it goes down every period;
   @FOR( PERIOD( T) | T #GT# 1:
      PRICE( 1, T) = PRICE( 1, T - 1) * DNF);
! Now compute for all other states S, period T;
   @FOR( STATE( S, T)| T #GT# 1 #AND# S #GT# 1:
      PRICE( S, T) = PRICE( S - 1, T - 1) * UPF);
! Set values in the final period;
    @FOR( PERIOD( S):
    VAL( S, LASTP)= @SMAX( PRICE( S, LASTP) - STRIKE,0)
        );
! Do the dynamic programming;
   @FOR( STATE( S, T) | T #LT# LASTP:
      VAL( S, T) = @SMAX( PRICE( S, T) - STRIKE,
             DISF * ( PUP * VAL( S + 1, T + 1) +
               ( 1 - PUP) * VAL( S, T + 1))));
! Finally, the value of the option now;
 VALUE = VAL( 1, 1);
END
```

The @SMAX function in the dynamic programming section corresponds to the decision in period T to either exercise the option and make an immediate profit of *PRICE*(*S*, *T*) – *STRIKE*, or wait (at least) until next period. If we wait until next period, the price can go up with probability *PUP* or down with probability 1- *PUP*. In either case, to convert next period's value to this period's value, we must multiply by the discount factor, *DISF*. The interesting part of the solution to this model gives:

```
Variable           Value
   VALUE          6.549348
```

The actual price of this option in the Wall Street Journal when there were 19 weeks until expiration was $6.625. So, it looks like this option is not a good buy if we are confident in our input data.

12.8.1 Binomial Tree Models of Interest Rates

Financial options based on interest rates are becoming widely available, just as options on stock prices have become widely available. In order to evaluate an interest rate option properly, we need a model of the random behavior of interest rates.

Interest rates behave differently than stock prices. Most notably, interest rates tend to hover in a finite interval (e.g., 2% to 20% per year); whereas, stock prices continue to increase year after year. Not surprisingly, a different model must be used to model interest rates. One of the simpler, yet realistic, methods for evaluating interest rate based options was developed by Black, Derman, and Toy (1990). Heath, Jarrow, and Morton (1992) present another popular model of interest rate movements.

The Black/Derman/Toy (BDT) model tries to fit two sets of data: a) the yield curve for bonds, and b) the volatility in the yield to maturity (*YTM*) for bonds. For a *T* period problem, the random variable of

interest is the forward interest rate in each period 1, 2, …, *T*. For period 1, the forward rate is known. For periods $t = 2, 3, \ldots, T$, the BDT model chooses t forward rates, so these rates are consistent with: a) the *YTM* curve, and b) the observed volatility in *YTM*. The BDT model assumes the probability of an increase in the interest rate in a period = probability of a decrease = .5. The possible rates in a period for the BDT model are determined by two numbers: a) a base rate, which can be thought of as chosen to match the mean *YTM*, and b) a rate ratio chosen to match the volatility in the *YTM*. Specifically, the BDT model assumes $r_{i+1,t} / r_{i,t} = r_{i,t} / r_{i-1,t}$ for the ith forward rate in period t. Thus, if $r_{1,t}$ and $r_{2,t}$ are specified in period t, then all the other rates for period t are determined.

Below is a LINGO implementation of the BDT model:

```
MODEL:
SETS:
! Black/Derman/Toy binomial interest rate model(BDTCALB);
! Calibrate it to a given yield curve and volatilities;
 PORM/1..5/:   ! (INPUTS:)For each maturity;
   YTM,    ! Yield To Maturity of Zero Coupon Bond;
   VOL;    ! Volatility of Yield To Maturity of ZCB;
 STATE( PORM, PORM)| &1 #GE# &2:
   FSRATE; ! (OUTPUT:)Future short rate in period j, state k;
ENDSETS
DATA:
 YTM = .08, .0812, .0816, .0818, .0814;
 VOL =   0, .1651, .1658, .1688, .1686;
! Write the forward rates to a file;
 @TEXT( 'forwrdr.dat') = FSRATE;
ENDDATA
!-------------------------------------------------;
SETS:
 TWO/1..2/:;
 VYTM( PORM, TWO): YTM2; ! Period 2 YTM's;
 MXS( PORM, PORM, PORM)|&1#GE# &2 #AND# &2 #GE# &3:
   PRICE; ! Price of a ZCB of maturity i, in period j,  state k;
ENDSETS
! Short rate ratios must be constant
      (Note: C/B=B/A <=> C=BB/A);
 @FOR( STATE( J, K)| K #GT# 2:
   FSRATE( J, K) =
FSRATE( J, K -1) * FSRATE( J, K-1)/ FSRATE( J, K-2);
    @FREE( FSRATE( J, K));
    );
! Compute prices for each maturity in each period and state;
@FOR( MXS( I, J, K)| J #EQ# I:
    @FREE( PRICE( I, I, K));
    PRICE( I, I, K) = 1/( 1 + FSRATE( I, K)); );
@FOR( MXS( I, J, K) | J #LT# I:
    @FREE( PRICE( I, J, K));
    PRICE( I, J, K) = .5 * ( PRICE(I, J + 1, K) + PRICE(I, J + 1, K +
1))/( 1 + FSRATE( J, K));
    );
!For each maturity, price in period 1 must be consistent with its YTM;
@FOR( PORM( I):
  PRICE( I, 1, 1)*( 1 + YTM( I))^I = 1;
  );
```

```
 ! Compute period 2 YTM's for each maturity;
@FOR( VYTM( I, K)| I #GT# 1:
  YTM2( I, K) = (1/ PRICE( I, 2, K)^(1/( I-1))) - 1;
   );
  ! Match the volatilities for each maturity;
@FOR( PORM( I)| I #GT# 1:
  .5 * @LOG( YTM2( I, 2)/ YTM2( I, 1)) = VOL( I);
   );
END
```

When solved, we get the following forward interest rates:

```
     Variable           Value
FSRATE( 1, 1)       0.8000000E-01
FSRATE( 2, 1)       0.6906015E-01
FSRATE( 2, 2)       0.9607968E-01
FSRATE( 3, 1)       0.5777419E-01
FSRATE( 3, 2)       0.8065491E-01
FSRATE( 3, 3)       0.1125973
FSRATE( 4, 1)       0.4706528E-01
FSRATE( 4, 2)       0.6690677E-01
FSRATE( 4, 3)       0.9511292E-01
FSRATE( 4, 4)       0.1352100
FSRATE( 5, 1)       0.3900926E-01
FSRATE( 5, 2)       0.5481167E-01
FSRATE( 5, 3)       0.7701470E-01
FSRATE( 5, 4)       0.1082116
FSRATE( 5, 5)       0.1520465
```

We can display these forward rates in a more intuitive tree form:

```
                        Period
   1          2           3          4          5
                                                0.1520482
                                     0.1352100  0.1082116
                          0.1125973  0.0951129  0.0770147
              0.0960797   0.0806549  0.0669068  0.0548117
 0.0800000    0.0690602   0.0577742  0.0470653  0.0390093
```

Thus, the BDT model implies that, at the start of period 1, the interest rate is .08. At the start of period 2 (end of period 1), the interest rate will be with equal probability, either .0690602 or .0960797, etc.

Now, let us suppose we want to compute the value of an interest rate cap of 10% in period 5. That is, we would like to buy insurance against the interest rate being greater than .10 in period 5. We see there are two possible interest rates, .1520482 and .1082116, that would cause the insurance to "kick in". Assuming interest is paid at the end of each period, it should be clear such a cap is worth either 0, .0082116, or .0520482 at the end of period 5. We can calculate the expected value in earlier periods with the following dynamic programming value tree:

		Period		
1	2	3	4	5
				.0520482
			.026294	.008212
		.013273	.003705	0
	.006747	.001692	0	0
.003440	.000783	0	0	0

These values apply to the end of each year. At the beginning of the first year, we would be willing to pay .003440/ 1.08 = .003189 per dollar of principal for this cap on interest rate at the end of year five. The following LINGO model will read in the *FSRATE* data computed by the previous model and compute the above table of values. Note the *FSRATE's* need be computed only once. It can then be used to evaluate or price various *CAP*'s, or caplets as they are sometimes known:

```
MODEL:
SETS:
! Black/Derman/Toy binomial interest rate model.
 Compute value of a cap.(BDTCMP);
 PORM/1..5/:  ;
 STATE( PORM, PORM)| &1 #GE# &2:
 FSRATE,
 !(OUTPUT:)Future short rate in period j,state k;
 VALUE;  !  Value of the option in this state;
ENDSETS
DATA:
  CAP = .10;
  FSRATE = @TEXT( forwrdr.dat);
ENDDATA
!-------------------------------------------------;
 LASTP = @SIZE( PORM);
 @FOR( PORM( K):
VALUE(LASTP, K) = @SMAX(0, FSRATE(LASTP, K) - CAP);
  );
 @FOR( STATE( J, K) | J #LT# LASTP:
   VALUE( J, K) =
.5 * (VALUE(J + 1, K + 1)/(1 + FSRATE(J + 1, K + 1))
      + VALUE( J + 1, K)/(1 + FSRATE( J + 1, K)));
     );
!  The value at the beginning of period 1;
  VALUE0 = VALUE( 1, 1)/( 1 + FSRATE( 1, 1));
END
```

12.8.2 Binomial Tree Models of Foreign Exchange Rates

DeRosa (1992) describes a simple binomial tree model of foreign exchange rates. The following LINGO model illustrates the valuation of an option on the German Mark when there were 36 days until its expiration. This model illustrates the case of an American style option. That is, the option may be exercised any time before its expiration. It is a simple matter to simplify the model to the case of a European style option, which can be exercised only at maturity.

```
MODEL:
SETS:       !(OPTONFX);
! Binomial option pricing model on foreign exchange:
 What is the value in $ of an option to buy one unit
 Of a foreign currency at specified/strike exchange
 rate?  The binomial model assumes the exchange rate
  can either go up from one period to the next by a
  fixed factor,  or down by another fixed factor;
! No. of discrete periods to use, including time now
  ( 6 means 5 future periods);
   PERIOD /1..6/:;
ENDSETS
DATA:
! Current exchange rate, $ per foreign unit;
   XCURR = .5893;
! Strike exchange rate, i.e., right to exchange
   $1 for one foreign unit at this rate;
   XSTRK =.58;
! Yearly interest rate in $ country;
   IRD = .0581;
! Yearly interest rate in foreign country;
   IRF = .0881;
! Years to maturity for the option;
   MATRT = .098630137; !( = 36/365);
! Yearly variance in exchange rate;
   SIG = .13;
ENDDATA
!-----------------------------------------------------;
SETS:
!Generate state matrix for the DP. STATE( S, T) may
be entered from STATE(S, T-1) if FX rate went down,
or from STATE( S - 1, T - 1) if FX rate went up;
 STATE( PERIOD, PERIOD)| &1 #LE# &2:
         FXRATE,  ! There is an FX rate, and...;
         VAL;     ! a value of the option;
ENDSETS
! Compute number of periods;
   LASTP = @SIZE( PERIOD);
! Initialize the FXRATE table;
   FXRATE( 1, 1) = XCURR;
! Compute some constants;
!   To avoid warning messages when IRDIFM < 0;
   @FREE( IRDIFM);
   IRDIFM = ( IRD - IRF) * MATRT/( LASTP - 1);
   SIGMSR = SIG * (( MATRT/( LASTP - 1))^.5);
   DISF = @EXP( - IRD * MATRT/( LASTP - 1));
```

```
! The up factor;
   UPF = @EXP( IRDIFM + SIGMSR);
! The down factor;
   DNF = @EXP( IRDIFM - SIGMSR);
! Probability of an up move( assumes SIG > 0);
   PUP =  (@EXP( IRDIFM)- DNF)/( UPF - DNF);
   PDN = 1 - PUP;
! First the states where it goes down every period;
   @FOR( PERIOD( T) | T #GT# 1:
      FXRATE( 1, T) = FXRATE( 1, T - 1) * DNF);
! Now compute for all other states S, period T;
   @FOR( STATE( S, T)| T #GT# 1 #AND# S #GT# 1:
      FXRATE( S, T) = FXRATE( S - 1, T - 1) * UPF);
! Do the dynamic programming;
! Set values in the final period;
   @FOR( PERIOD( S):
     VAL( S, LASTP) =
              @SMAX( FXRATE( S, LASTP) - XSTRK, 0));
! and for the earlier periods;
   @FOR( STATE( S, T) | T #LT# LASTP:
      VAL( S, T) = @SMAX( FXRATE( S, T) - XSTRK,
                 DISF * ( PUP * VAL( S + 1, T + 1) +
                               PDN * VAL( S, T + 1))));
! Finally, the value of the option now;
 VALUE = VAL( 1, 1);
END
```

It is of interest to look at all of the states computed by the model:

```
      Variable           Value
         XCURR       0.5893000
         XSTRK       0.5800000
           IRD       0.5810000E-01
           IRF       0.8810000E-01
         MATRT       0.9863014E-01
           SIG       0.1300000
         LASTP        6.000000
        IRDIFM      -0.5917808E-03
        SIGMSR       0.1825842E-01
          DISF       0.9988546
           UPF        1.017824
           DNF       0.9813264
           PUP       0.4954355
           PDN       0.5045645
         VALUE       0.1393443E-01
FXRATE( 1, 1)       0.5893000
FXRATE( 1, 2)       0.5782956
FXRATE( 1, 3)       0.5674967
FXRATE( 1, 4)       0.5568995
FXRATE( 1, 5)       0.5465002
FXRATE( 1, 6)       0.5362950
FXRATE( 2, 2)       0.5998035
FXRATE( 2, 3)       0.5886029
FXRATE( 2, 4)       0.5776116
```

```
FXRATE( 2, 5)       0.5668255
FXRATE( 2, 6)       0.5562408
FXRATE( 3, 3)       0.6104941
FXRATE( 3, 4)       0.5990940
FXRATE( 3, 5)       0.5879067
FXRATE( 3, 6)       0.5769283
FXRATE( 4, 4)       0.6213753
FXRATE( 4, 5)       0.6097720
FXRATE( 4, 6)       0.5983853
FXRATE( 5, 5)       0.6324505
FXRATE( 5, 6)       0.6206403
FXRATE( 6, 6)       0.6437230
   VAL( 1, 1)       0.1393443E-01
   VAL( 1, 2)       0.6976915E-02
   VAL( 1, 3)       0.2228125E-02
   VAL( 1, 4)       0.0000000
   VAL( 1, 5)       0.0000000
   VAL( 1, 6)       0.0000000
   VAL( 2, 2)       0.2105240E-01
   VAL( 2, 3)       0.1182936E-01
   VAL( 2, 4)       0.4502463E-02
   VAL( 2, 5)       0.0000000
   VAL( 2, 6)       0.0000000
   VAL( 3, 3)       0.3049412E-01
   VAL( 3, 4)       0.1931863E-01
   VAL( 3, 5)       0.9098311E-02
   VAL( 3, 6)       0.0000000
   VAL( 4, 4)       0.4137534E-01
   VAL( 4, 5)       0.2977199E-01
   VAL( 4, 6)       0.1838533E-01
   VAL( 5, 5)       0.5245049E-01
   VAL( 5, 6)       0.4064034E-01
   VAL( 6, 6)       0.6372305E-01
```

Thus, the value of this option is *VAL* (1, 1) = \$0.01393443. For example, the option to buy 100 Marks for \$58 any time during the next 36 days is worth \$1.393443. The actual option on which the above was based had a price of \$1.368 per 100 Marks. The actual option also happened to be a European style option, rather than American. An American option can be exercised at any point during its life. A European option can be exercised only at its maturity. Thus, it is not surprising the above model should attribute a higher value to the option.

12.9 Decisions Under Uncertainty with an Infinite Number of Periods

We can consider the case of an infinite number of periods if we have a system where:

a) we can represent the state of the system as one of a finite number of possible states,
b) we can represent our possible actions as a finite set,
c) given that we find the system in state s and take action x in a period, nature moves the system to state j the next period with probability $p(x, j)$,
d) a cost $c(s,x)$ is incurred when we take action x from state s.

Such a system is called a Markov decision process and is, in fact, quite general. Our goal is to find the best action to take for each state to minimize the average cost per period. Puterman (1994) provides an excellent introduction to applications of Markov Decision Processes, as well as a thorough presentation of the theory. Manne (1960) showed how to formulate the problem of determining the best action for each state as a linear program. He defined:

$w(s,x)$ = probability that in the steady state the state is s and we take action x.

This implicitly allows for randomized policies. That is, the decision maker could flip a coin to determine his decision. It turns out, however, that there is always an optimal policy that is deterministic. Allowing randomized policies is simply a convenient computational approach.

Manne's LP is then:

$$\min = \sum_{s,x} c(s, x)\, w(s, x)$$

subject to:

$$\sum_{s,x} w(s, x) = 1,$$

For each state s:

$$\sum_{x} w(s, x) = \sum_{r,x} w(r, x)\, p(x, s).$$

Notice the probability of going to state s depends only upon the action taken, x. Some descriptions of Markov Decision Processes give an apparently more general definition of the state transition process by letting the probability of state s depend not only upon the decision x, but also the previous state r. Thus, the transition matrix would be a three dimensional array, $p(r, x,s)$. By giving a suitably general definition of "decision", however, the format where the next state depends only upon the current decision can represent any situation representable with the three dimensional notation. For many practical problems, the $p(x,s)$ notation is more natural. For example, in an inventory system, if we decide to raise the inventory level to 15, the probability that the next state is 7 is usually independent of whether we raised the inventory level to 15 from an inventory level of 5 or of 6. Similarly, in a maintenance system, if we completely overhaul the system, the probability of the next state should be independent of the state before the overhaul. Another way of thinking about a decision is that it chooses the probability distribution from which nature chooses the next state.

Wang and Zaniewski (1996) describe a system based on a Markov decision process model for scheduling maintenance on highways in Arizona and a number of other states. It has been in use since 1982. A state in this application corresponds to a particular condition of a section of roadway. Transition probabilities describe the statistical manner in which a road deteriorates. Actions correspond to possible road repairs, such as patch, resurface, or completely replace. Electrical distribution companies have similar maintenance problems. With time, tree branches near power lines get longer and equipment deteriorates. The actions available to the electrical distribution company are things like tree trimming, installing squirrel guards, replacing old equipment, etc.

12.9.1 Example: Cash Balance Management

Suppose we are managing a cash account for which each evening there is a random input or output of cash as revenue arrives and/or bills get paid. Each morning, we observe the account level. If the cash level gets too high, we want to transfer some of the cash to a longer term investment account that pays a higher interest rate. However, if the cash account gets too low, we want to transfer funds from a longer term account into the cash account, so we always have sufficient cash on hand. Because we require discrete scenarios, let us represent the cash-on-hand status as multiples of $1000. In order to avoid negative subscripts, let us make the following correspondence between cash on hand and state:

Cash on hand:	–2000	–1000	0	1000	2000	3000	4000	5000
State:	1	2	3	4	5	6	7	8
Cost:	14	7	0	2	4	6	8	10

Given a state, we can move to any other state by transferring funds if we incur:

1) a fixed cost of $3 for making any transfer, and
2) a variable cost of $5 per thousand dollars transferred.

Further, suppose that over night only three transitions are possible: go down one state, stay put, or go up one state. Their probabilities are: Prob{down one state} = .4; Prob{no change} = .1; Prob{up one state} = .5.

In state 1, we assume the probability of no change is .5; whereas, in state 8, the probability of no change is .6. We can think of the sequence of events each day as:

i. we observe the cash level in the morning,
ii. we make any transfers deemed appropriate,
iii. overnight the cash level either increases by $1000, stays the same, or decreases by $1000.

A "scalar" model is:

```
MIN = 10 * W88 + 18 * W87 + 23 * W86 + 28 * W85
+ 33 * W84 + 38 * W83 + 43 * W82 + 48 * W81 + 16 * W78
+  8 * W77 + 16 * W76 + 21 * W75 + 26 * W74 + 31 * W73
+ 36 * W72 + 41 * W71 + 19 * W68 + 14 * W67 +  6 * W66
+ 14 * W65 + 19 * W64 + 24 * W63 + 29 * W62 + 34 * W61
+ 22 * W58 + 17 * W57 + 12 * W56 +  4 * W55 + 12 * W54
+ 17 * W53 + 22 * W52 + 27 * W51 + 25 * W48 + 20 * W47
+ 15 * W46 + 10 * W45 +  2 * W44 + 10 * W43 + 15 * W42
+ 20 * W41 + 28 * W38 + 23 * W37 + 18 * W36 + 13 * W35
+  8 * W34 +  8 * W32 + 13 * W31 + 40 * W28 + 35 * W27
+ 30 * W26 + 25 * W25 + 20 * W24 + 15 * W23 +  7 * W22
+ 15 * W21 + 52 * W18 + 47 * W17 + 42 * W16 + 37 * W15
+ 32 * W14 + 27 * W13 + 22 * W12 + 14 * W11;
! Probabilities sum to 1;
    W88 + W87 + W86 + W85 + W84 + W83  + W82 + W81
  + W78  + W77  + W76  + W75  + W74  + W73  + W72  + W71
  + W68  + W67  + W66  + W65  + W64  + W63  + W62  + W61
  + W58  + W57  + W56  + W55  + W54  + W53  + W52  + W51
  + W48  + W47  + W46  + W45  + W44  + W43  + W42  + W41
```

```
  + W38  + W37  + W36  + W35  + W34  + W33  + W32  + W31
  + W28  + W27  + W26  + W25  + W24  + W23  + W22  + W21
  + W18  + W17  + W16  + W15  + W14  + W13  + W12  + W11 = 1;
! Prob{out of state 1}- Prob{ into state 1} = 0;
- .4 * W82 - .5 * W81 - .4 * W72 - .5 * W71 - .4 * W62
- .5 * W61 - .4 * W52 - .5 * W51 - .4 * W42 - .5 * W41
- .4 * W32 - .5 * W31 - .4 * W22 - .5 * W21 + W18 + W17
+ W16 + W15 + W14 + W13 + .6 * W12 + .5 * W11 = 0;
! Into state 2;
- .4 * W83 - .1 * W82 - .5 * W81 - .4 * W73 - .1 * W72
- .5 * W71 - .4 * W63 - .1 * W62 - .5 * W61 - .4 * W53
- .1 * W52 - .5 * W51 - .4 * W43 - .1 * W42 - .5 * W41
- .4 * W33 - .1 * W32 - .5 * W31 + W28 + W27 + W26 + W25 + W24 + .6 *
W23 + .9 * W22 + .5 * W21 - .4 * W13 - .1 * W12 - .5 * W11 = 0;
! Into state 3;
- .4 * W84 - .1 * W83 - .5 * W82 - .4 * W74 - .1 * W73
- .5 * W72 - .4 * W64 - .1 * W63 - .5 * W62 - .4 * W54
- .1 * W53 - .5 * W52 - .4 * W44 - .1 * W43 - .5 * W42 + W38 + W37 +
W36 + W35 + .6 * W34 + .9 * W33 + .5 * W32 + W31 - .4 * W24 - .1 * W23
- .5 * W22 - .4 * W14 - .1 * W13 - .5 * W12 = 0;
! Into state 4;
- .4 * W85 - .1 * W84 - .5 * W83 - .4 * W75 - .1 * W74
- .5 * W73 - .4 * W65 - .1 * W64 - .5 * W63 - .4 * W55
- .1 * W54 - .5 * W53 + W48 + W47 + W46 + .6 * W45 + .9 * W44 + .5 *
W43 + W42 + W41 - .4 * W35 - .1 * W34 - .5 * W33 - .4 * W25 - .1 * W24
- .5 * W23 - .4 * W15 - .1 * W14 - .5 * W13 = 0;
! Into state 5;
- .4 * W86 - .1 * W85 - .5 * W84 - .4 * W76 - .1 * W75
- .5 * W74 - .4 * W66 - .1 * W65 - .5 * W64 + W58 + W57
+ .6 * W56 + .9 * W55 + .5 * W54 + W53 + W52 + W51 - .4 * W46 - .1 *
W45 - .5 * W44 - .4 * W36 - .1 * W35 - .5 * W34 - .4 * W26 - .1 * W25
- .5 * W24 - .4 * W16 - .1 * W15 - .5 * W14 =0;
! Into state 6;
- .4 * W87 - .1 * W86 - .5 * W85 - .4 * W77 - .1 * W76
- .5 * W75 + W68 + .6 * W67 + .9 * W66 + .5 * W65 + W64 + W63 + W62  +
W61 - .4 * W57 - .1 * W56 - .5 * W55 - .4 * W47 - .1 * W46 - .5 * W45
- .4 * W37 - .1 * W36 - .5 * W35 - .4 * W27 - .1 * W26 - .5 * W25 - .4
* W17 - .1 * W16 - .5 * W15 = 0;
! Into state 7;
- .4 * W88 - .1 * W87 - .5 * W86 + .6 * W78 + .9 * W77 + .5 * W76 + W75
+ W74 + W73 + W72 + W71 - .4 * W68 - .1 * W67 - .5 * W66 - .4 * W58 -
.1 * W57 - .5 * W56 - .4 * W48 - .1 * W47 - .5 * W46 - .4 * W38 - .1 *
W37 - .5 * W36 - .4 * W28 - .1 * W27 - .5 * W26 - .4 * W18 - .1 * W17
- .5 * W16 = 0;
! Into state 8;
.4 * W88 + .5 * W87 + W86 + W85 + W84 + W83 + W82 + W81 - .6 * W78 -
.5 * W77 - .6 * W68 - .5 * W67 - .6 * W58 - .5 * W57 - .6 * W48 - .5 *
W47 - .6 * W38 - .5 * W37 - .6 * W28 - .5 * W27 - .6 * W18 - .5 * W17
= 0;
END
```

Note in the objective, the term 23 * *W86* can be thought of as (10 + 3 + 5 * 2)**W86*. Similarly, the term + .6 * *W12* in the "into state 1" constraint, comes from the fact that the probability there is a change

out of state 1 in the morning is *W12*. At the same time, there is also a probability of changing into state 1 from state 1 the previous morning of *W12* * Prob{down transition over night} = *W12**.4. The net is *W12* - .4**W12* = .6**W12*.

Part of the solution report is reproduced below:

```
Obj. value=     5.633607
Variable           Value        Reduced Cost
     W64        0.1024590           0.0000000
     W55        0.2049180           0.0000000
     W44        0.2663934           0.0000000
     W22        0.1311475           0.0000000
     W13        0.5245902E-01       0.0000000
     W33        0.2426230           0.0000000
```

For example, variable *W64* = 0.1024590 means that, in a fraction 0.1024590 of the periods, we will find the system in state 6 and we will (or should) take action 4. Note, there is no other positive variable involving state 6. So, this implies, if the system is in state 6, we should always take action 4. The expected cost per day of this policy is 5.633607.

Summarizing:
If the state is 1 or less, we should raise it to state 3.
If the state is 6 or more, we should drop it to state 4.
If the state is 2, 3, 4, or 5, we should stay put.

Here is a general purpose sets formulation of a Markov decision problem, with data specific to our cash balance problem:

```
SETS: ! Markov decision process model(MARKOVDP);
 STATE: H;
 DCSN:;
 SXD( STATE, DCSN): C, W;
 DXS( DCSN, STATE): P;
ENDSETS
DATA:
! Data for the cash balance problem;
! The states ....;
STATE= SN2K SN1K S000 SP1K SP2K SP3K SP4K SP5K;
! The cost of finding system in a given state;
   H =  14   7    0    2    4    6    8   10;
! Possible decisions;
 DCSN= DN2K DN1K D000 DP1K DP2K DP3K DP4K DP5K;
! The cost of explicitly changing to any other state;
   C =  0    8   13   18   23   28   33   38
        8    0    8   13   18   23   28   33
       13    8    0    8   13   18   23   28
       18   13    8    0    8   13   18   23
       23   18   13    8    0    8   13   18
       28   23   18   13    8    0    8   13
       33   28   23   18   13    8    0    8
       38   33   28   23   18   13    8    0;
```

```
! Prob{ nature moves system to state j| we made decision i};
   P = .5   .5    0    0    0    0    0    0
       .4   .1   .5    0    0    0    0    0
       0    .4   .1   .5    0    0    0    0
       0     0   .4   .1   .5    0    0    0
       0     0    0   .4   .1   .5    0    0
       0     0    0    0   .4   .1   .5    0
       0     0    0    0    0   .4   .1   .5
       0     0    0    0    0    0   .4   .6;
ENDDATA
!-----------------------------------------------------;
!Minimize the average cost per period;
MIN=@SUM(SXD( S, X): ( H( S) + C( S, X))* W( S, X));
!The probabilities must sum to 1;
  @SUM( SXD( S, X): W( S, X)) = 1;
!Rate at which we exit state S = rate of entry to S.
 Note, W( S, X) = Prob{ state is S and we make decision X};
  @FOR( STATE( S):
   @SUM( DCSN( X): W( S, X))=
    @SUM( SXD( R, K): W( R, K)* P( K, S));
            );
```

In the above example, the number of decision alternatives equaled the number of states, so the transition matrix was square. In general, the number of decisions might be more or less than the number of states, so the transition matrix need not be square.

The above model minimizes the average cost per period in the long run. If discounted present value, rather than average cost per period, is of concern, then see d'Epenoux (1963) for a linear programming model, similar to the above, that does discounting.

12.10 Chance-Constrained Programs

A drawback of the methods just discussed is problem size can grow very large if the number of possible states of nature is large. Chance-constrained programs do not apply to exactly the same problem and, as a result, do not become large as the number of possible states of nature gets large. The stochastic programs discussed thus far had the feature that every constraint had to be satisfied by some combination of first- and second-period decisions. Chance-constrained programs, however, allow each constraint to be violated with a certain specified probability. An advantage to this approach for tolerating uncertainty is the chance-constrained model has essentially the same size as the LP for a corresponding problem with no random elements.

We will illustrate the idea with the snow removal problem. Under the chance-constrained approach, there are no second-stage decision variables, and we would have to specify a probability allowance for each constraint. For example, we might specify that with probability at least 0.75 we must be able to provide the snow removal capacity required by the severity of the winter. For our problem, it is very easy to see that this means we must provide 5,100 truck-days of snow removal capacity. For example, if only 4,400 truck-days of capacity were provided, then the probability of sufficient capacity would only be 0.4. Let us assume one truck-day of operation costs \$116, and one truck-day of salting equals 1.14 truck-days of plowing. Then, the appropriate chance-constrained LP is the simple model:

```
Min=70*BF1 + 20*BS1 + 116*P    + 116*S;
      -BF1               + P        + S =    0;
               -BS1                 + S =    0;
                          P         + S >= 5000;
                          P + 1.14 * S >= 5100;
```

12.11 Problems

1. What is the expected value of perfect information in the corn/soybean/sorghum planting problem?
2. The farmer in the corn/soybean/sorghum problem is reluctant to plant all soybeans because, if the season is wet, he will make \$20 less per acre than he would if he had planted all corn. Can you react to his risk aversion and recommend a planting *mix* where the profit per acre is never more than \$15 from the planting mix that in retrospect would have been best for the particular outcome?
3. Analyze the snow removal problem of this chapter for the situation where the cost of fuel in a cold winter is \$80 per truck-day rather than \$73, and the cost of salt in a cold winter is \$35 rather than \$32. Include in your analysis the derivation of the expected value of perfect information.
4. A farmer has 1000 acres available for planting to either corn, sorghum, or soybeans. The yields of the three crops, in bushels per acre, as a function of the two possible kinds of season are:

	Corn	Sorghum	Beans
Wet	100	43	45
Dry	45	35	33

The probability of a wet season is 0.6. The probability of a dry season is 0.4. Corn sells for \$2/bushel; whereas, sorghum and beans each sell for \$4/bushel. The total production cost for any crop is \$100/acre, regardless of type of season. The farmer can also raise livestock. One unit of livestock uses one hundred bushels of corn. The profit contribution of one unit of livestock, exclusive of its corn consumption, is \$215. Corn can be purchased at any time on the market for \$2.20/bushel. The decision of how much to raise of livestock and of each crop must be made before the type of season is known.

a) What should the farmer do?

b) Formulate and solve the problem by the scenario-based stochastic programming approach.

5. A firm serves essentially two markets, East and West, and is contemplating the location of one or more distribution centers (DC) to serve these markets. A complicating issue is the uncertainty in demand in each market. The firm has enumerated three representative scenarios to characterize the uncertainty. The table below gives (i) the fixed cost per year of having a DC at each of three candidate locations, and (ii) the profit per year in each market as a function of the scenario and which DC is supplying the market. Each market will be assigned to that one open DC that results in the most profit. This assignment can be done after we realize the scenario that holds. The DC location decision must be made before the scenario is known.

Profit by Scenario/Region and Supplier DC

		Scenario One		Scenario Two		Scenario Three	
DC Location	Fixed Cost	East	West	East	West	East	West
A	51	120	21	21	40	110	11
B	49	110	28	32	92	70	70
C	52	60	39	20	109	20	88

For example, if Scenario Three holds and we locate DC's at *A* and *C*, East would get served from *A*, West from *C*, and total profits would be 110 + 88 − 51 − 52 = 95.

a) If Scenario One holds, what is the best combination of DC's to have open?

b) If Scenario Two holds, what is the best combination of DC's to have open?

c) If Scenario Three holds, what is the best combination of DC's to have open?

d) If all three scenarios are equally likely, what is the best combination of DC's to have open?