Optimization Methods in Finance

Instructor:
Gerard Cornuejols
Graduate School of Industrial Administration
Carnegie Mellon University, Pittsburgh, PA 15213 USA

Spring 2003

Objectives of the Course

Optimization Methods in Finance is a project based course in which you learn how to apply optimization techniques to solve financial problems. The course will cover optimization techniques such as linear programming, nonlinear programming, integer programming, dynamic programming, stochastic programming and robust optimization with examples of financial applications. There will be two projects, both involving real world data (with minor simplifications) of problems faced by practitioners. The first project will be a bond portfolio construction project for a small pension fund. The second project will be a group project done in teams of three students. Four topics will be proposed in class: a portfolio optimization problem, constructing an index fund, structuring collaterized mortgage obligations, constructing a synthetic option. For each topic, each team will write a two- to three-page "proposal". Based on these proposals, each team will be assigned one topic as second project. Each team will present the results of their analysis in class.

Instructor

Gérard Cornuéjols, GSIA 232A, phone 268-2284, fax 268-7357
email: gc0v@andrew.cmu.edu
secretary: Barbara Carlson 268-1342, GSIA 232.

Teaching Assistants

Atul Bhandari, GSIA 211, phone 268-6895
email: atv1@andrew.cmu.edu

Luis Zuluaga, GSIA 205, phone 268-2463
email: lzuluaga@andrew.cmu.edu

Miroslav Karamanov, GSIA 203, phone 268-5742
email: miroslav@andrew.cmu.edu

Software

The software for this course will be Excel Solver or Premium Solver, Matlab and possibly other packages as the need arises.

Schedule

Date	Topic	Project
Week 1	Linear Programming Dedicated Bond Portfolio	 Project 1
Week 2	Nonlinear Programming Portfolio Optimization	 Project 2: Topic 1
Week 3	Integer Programming Constructing an Index Fund	**Project 1 due** Project 2: Topic 2
Week 4	Dynamic Programming Structuring CMO's	 Project 2: Topic 3
Week 5	Stochastic Programming Option Pricing	 Project 2: Topic 4
Week 6	Dynamic Portfolio Optimization Robust Optimization	
Week 7	Student Presentations	**Project 2 due**

Chapter 1

Linear Programming and Asset/Liability Cash Flow Matching

1.1 Short Term Financing

Corporations routinely face the problem of financing short term cash commitments. Linear programming can help in figuring out an optimal combination of financial instruments (hundreds of candidates are typically to be considered, such as bonds with various maturities) to meet these commitments. To illustrate this, consider the following problem. For simplicity of exposition, we keep the example very small.

A company has the following short term financing problem ($1000).

Month	J	F	M	A	M	J
Net Cash Flow	-150	-100	200	-200	50	300

The company has the following sources of funds

- A line of credit of up to $100 at an interest rate of 1% per month,
- It can issue 90-day commercial paper bearing a total interest of 2% for the 3-month period,
- Excess funds can be invested at an interest rate of 0.3% per month.

There are many questions that the company might want to answer. What interest payments will the company need to make between January and June? Is it economical to use the line of credit in some of the months? If so, when? How much? Linear programming gives us a mechanism for answering these questions quickly and easily. It also allows to answer some "what if" questions about changes in the data without having to resolve the problem. What if Net Cash Flow in January were -200 (instead of -150). What if the limit on the credit line were increased from 100 to 200. What if the negative Net Cash Flow in January is due to

the purchase of a machine worth 150 and the vendor allows part or all of the payment on this machine to be made in June at an interest of 3% for the 5-month period. The answers to these questions are readily available when this problem is formulated and solved as a linear program.

There are three steps in applying linear programming: modeling, solving, and interpreting.

1.1.1 Modeling

We begin by modeling the above short term financing problem. That is, we write it in the language of linear programming. There are rules about what you can and cannot do within linear programming. These rules are in place to make certain that the remaining steps of the process (solving and interpreting) can be successful.

Key to a linear program are the *decision variables*, *objective*, and *constraints*.

Decision Variables. The decision variables represent (unknown) decisions to be made. This is in contrast to *problem data*, which are values that are either given or can be simply calculated from what is given. For the short term financing problem, there are several possible choices of decision variables. We will use the following decision variables: the amount x_i drawn from the line of credit in month i, the amount y_i of commercial paper issued in month i, the excess funds z_i in month i and the company's wealth v in June. Note that, alternatively, one could use the decision variables x_i and z_i only, since excess funds and company's wealth can be deduced from these variables.

Objective. Every linear program has an objective. This objective is to be either minimized or maximized. This objective has to be *linear* in the decision variables, which means it must be the sum of constants times decision variables. $3x_1 - 10x_2$ is a linear function. x_1x_2 is not a linear function. In this case, our objective is simply to maximize v.

Constraints. Every linear program also has constraints limiting feasible decisions. Here we have three types of constraints: cash inflow = cash outflow for each month, upper bounds on x_i and nonnegativity of the decision variables x_i, y_i and z_i.

For example, in January ($i = 1$), there is a cash requirement of \$150. To meet this requirement, the company can draw an amount x_1 from its line of credit and issue an amount y_1 of commercial paper. Considering the possibility of excess funds z_1 (possibly 0), the cash flow balance equation is as follows.

$$x_1 + y_1 - z_1 = 150$$

Next, in February ($i = 2$), there is a cash requirement of \$100. In addition, principal plus interest of $1.01x_1$ is due on the line of credit and $1.003z_1$ is received on the invested excess funds. To meet the requirement in February, the company can draw an amount x_2 from its line of credit and issue an amount y_2 of commercial paper. So, the cash flow balance equation for February is as follows.

$$x_2 + y_2 - 1.01x_1 + 1.003z_1 - z_2 = 100$$

Similarly, for March, April, May and June, we get the following equations.

$$\begin{array}{rcrcrcrcrcr} x_3 & + & y_3 & - & 1.01x_2 & + & 1.003z_2 & - & z_3 & = & -200 \\ x_4 & - & 1.02y_1 & - & 1.01x_3 & + & 1.003z_3 & - & z_4 & = & 200 \\ x_5 & - & 1.02y_2 & - & 1.01x_4 & + & 1.003z_4 & - & z_5 & = & -50 \\ & - & 1.02y_3 & - & 1.01x_5 & + & 1.003z_5 & - & v & = & -300 \end{array}$$

Note that x_i is the balance on the credit line in month i, not the incremental borrowing in month i. Similarly, z_i represents the overall excess funds in month i. This choice of variables is quite convenient when it comes to writing down the upper bound and nonnegativity constraints.

$$\begin{array}{rcl} 0 & \leq & x_i \leq 100 \\ y_i & \geq & 0 \\ z_i & \geq & 0. \end{array}$$

Final Model. This gives us the complete model of this problem:

$$\begin{array}{lrcrcrcrcrcr} \max & v & & & & & & & & & & \\ & x_1 & + & y_1 & & & & & & - & z_1 & = & 150 \\ & x_2 & + & y_2 & - & 1.01x_1 & + & 1.003z_1 & - & z_2 & = & 100 \\ & x_3 & + & y_3 & - & 1.01x_2 & + & 1.003z_2 & - & z_3 & = & -200 \\ & x_4 & - & 1.02y_1 & - & 1.01x_3 & + & 1.003z_3 & - & z_4 & = & 200 \\ & x_5 & - & 1.02y_2 & - & 1.01x_4 & + & 1.003z_4 & - & z_5 & = & -50 \\ & & - & 1.02y_3 & - & 1.01x_5 & + & 1.003z_5 & - & v & = & -300 \\ & x_1 & \leq & 100 & & & & & & & & \\ & x_2 & \leq & 100 & & & & & & & & \\ & x_3 & \leq & 100 & & & & & & & & \\ & x_4 & \leq & 100 & & & & & & & & \\ & x_5 & \leq & 100 & & & & & & & & \\ & x_i, y_i, z_i & \geq & 0. & & & & & & & & \end{array}$$

Formulating a problem as a linear program means going through the above process to clearly define the decision variables, objective, and constraints.

1.1.2 Solving the Model with SOLVER

Special computer programs can be used to find solutions to linear programming models. The most widespread program is undoubtedly SOLVER, included in all recent versions of the Excel spreadsheet program. SOLVER, while not a state of the art code (which can cost upwards of $15,000 per copy) is a reasonably robust, easy-to-use tool for linear programming. SOLVER uses standard spreadsheets together with an interface to define variables, objective, and constraints.

Here are a brief outline and some hints and shortcuts on how to create a SOLVER spreadsheet:

- Start with a spreadsheet that has all of the data entered in some reasonably neat way.

  In the short term financing example, the spreadsheet might contain the cash flows, interest rates and credit limit.

- The model will be created in a separate part of the spreadsheet. Identify one cell with each decision variable. SOLVER will eventually put the optimal values in these cells.

  In the short term financing example, we could associate cells `$B$2` to `$B$6` with variables x_1 to x_5 respectively, cells `$C$2` to `$C$4` with the y_i variables, cells `$D$2` to `$D$6` with the z_i variables and, finally, `$E$2` with the variable v.

- A separate cell represents the objective. Enter a formula that represents the objective.

  For the short term financing example, we might assign cell `$B$8` to the objective function. Then, in cell `$B$8`, we enter the function `= $E$2`.

  This formula must be a linear formula, so, in general, it must be of the form: `cell1*cell1' + cell2*cell2' + ...`, where `cell1, cell2` and so on contain constant values and `cell1', cell2'` and so on are the decision variable cells.

- We then have a cell to represent the left hand side of each constraint (again a linear function) and another cell to represent the right hand side (a constant).

  In the short term financing example, cells `$B$10` to `$B$15` might contain the amounts generated through financing, for each month, and cells `$D$10` to `$D$15` the cash requirements for each month. For example, cell `$B$10` would contain the function `= $C$2 + $B$2 -$D$2` and cell `$D$10` the value 150. Similarly, rows 16 to 20 could be used to write the credit limit constraints.

  Helpful Hint: Excel has a function `sumproduct()` that is designed for linear programs. `sumproduct(a1..a10,b1..b10)` is identical to `a1*b1+a2*b2+a3*b3+...+a10*b10`. This function can save much time and aggravation. All that is needed is that the length of the first range be the same as the length of the second range (so one can be horizontal and the other vertical).

  Helpful Hint: It is possible to assign names to cells and ranges (under the `Insert-Name` menu). Rather than use `a1..a10` as the variables, you can name that range `var` (for example) and then use `var` wherever `a1..a10` would have been used.

- We then select `Solver` under the `Tools` menu. This gives a form to fill out to define the linear program.

- In the ``Set Cell'' box, select the `objective` cell. Choose `Maximize` or `Minimize`.

- In the ``By Changing Cells'', put in the range containing the variable cells.

- We next add the constraints. Press the ``Add...'' button to add constraints. The dialog box has three parts for the left hand side, the type of constraint, and the right hand side. Put the cell references for a constraint in the form, choose the right type, and press ``Add''. Continue until all constraints are added. On the final constraint, press ``OK''.

  Helpful Hint: It is possible to include ranges of constraints, as long as they all have the same type. `c1..e1 <= c3..e3` means `c1 <= c3`, `d1 <= d3`, `e1 <= e3`. `a1..a10 >= 0` means each individual cell must be greater than or equal to 0.

- Push the `options` button and toggle the ‘‘`Assume Linear Model`’’ in the resulting dialog box. This tells Excel to call a linear rather than a nonlinear programming routine so as to solve the problem more efficiently. This also gives you sensitivity ranges, which are not available for nonlinear models.

  Note that, if you want your variables to assume nonnegative values only, you need to specify this in the options box (alternatively, you can add nonegativity constraints in the previous step, in your constraints).

- Push the `Solve` button. In the resulting dialog box, select ‘‘`Answer`’’ and ‘‘`Sensitivity`’’. This will put the answer and sensitivity analysis in two new sheets. Ask Excel to ‘‘`Keep Solver values`’’, and your worksheet will be updated so that the optimal values are in the variable cells.

1.1.3 Interpreting the output of SOLVER

The ‘‘`Answer`’’ report looks as follows.

Target Cell (Max)

Cell	Name	Original Value	Final Value
$B$8	Objective	0	92.49694915

Adjustable Cells

Cell	Name	Original Value	Final Value
$B$2	$x1$	0	0
$B$3	$x2$	0	50.98039216
$B$4	$x3$	0	0
$B$5	$x4$	0	0
$B$6	$x5$	0	0
$C$2	$y1$	0	150
$C$3	$y2$	0	49.01960784
$C$4	$y3$	0	203.4343636
$D$2	$z1$	0	0
$D$3	$z2$	0	0
$D$4	$z3$	0	351.9441675
$D$5	$z4$	0	0
$D$6	$z5$	0	0
$E$2	v	0	92.49694915

Constraints

Cell	Name	Cell Value	Formula	Slack
$B$10	january	150	$B$10 = $D$10	0
$B$11	february	100	$B$11 = $D$11	0
$B$12	march	−200	$B$12 = $D$12	0
$B$13	april	200	$B$13 = $D$13	0
$B$14	may	−50	$B$14 = $D$14	0
$B$15	june	−300	$B$15 = $D$15	0
$B$16	x1limit	0	$B$16 <= $D$16	100
$B$17	x2limit	50.98039216	$B$17 <= $D$17	49.01960784
$B$18	x3limit	0	$B$18 <= $D$18	100
$B$19	x4limit	0	$B$19 <= $D$19	100
$B$20	x5limit	0	$B$20 <= $D$20	100

This report is fairly easy to read: the company's wealth v in june will be \$92,497. This is reported in `Final Value` of the `Objective` (recall that our units are in \$1000). To acheive this, the company will issue \$150,000 in commercial paper in january, \$49,000 in february and \$203,400 in march. In addition, it will draw \$50,980 from its line of credit in february. Excess cash of \$351,944 in march will be invested for just one month. All this is reported in the `Adjustable Cells` section of the report. For this particular application, the `Constraints` section of the report does not contain anything useful. On the other hand, very useful information can be found in the sensitivity report. This will be discussed in Section 3.

1.1.4 Modeling Languages

Linear programs can be formulated using modeling languages such as AMPL, GAMS or OPL. The need for these modeling languages arises because the Excel spreadsheet format becomes inadequate when the size of the linear program increases. A modeling language lets people use common notation and familiar concepts to formulate optimization models and examine solutions. Most importantly, large problems can be formulated in a compact way. Once the problem has been formulated using a modeling language, it can be solved using any number of solvers. A user can switch between solvers with a single command and select options that may improve solver performance. In these notes, I will sometimes present formulations using a modeling language format since it is easier to follow than an Excel Solver formulation. The short term financing model would be formulated as follows (all variables are assumed to be nonnegative unless otherwise specified).

```
DATA
LET T=6 be the number of months to plan for
L(t) = Liability in month t=1,...,T
ratex = monthly interest rate on line of credit
ratey = 3-month interest rate on commercial paper
ratez = monthly interest rate on excess funds
VARIABLES
```

```
x(t) = Amount drawn from line of credit in month t
y(t) = Amount of commercial paper issued in month t
z(t) = Excess funds in month t
OBJECTIVE (Maximize wealth in June)
Max z(6)
CONSTRAINTS
Month(t=1:T): x(t) - (1+ratex)*x(t-1) + y(t) - (1+ratey)*y(t-3)
-z(t) +(1+ratez)*z(t-1) = L(t)
Month(t=1:T-1):  x(t) < 100
Boundary conditions on x:  x(0)=x(6) =0
Boundary conditions on y:  y(-2)=y(-1)=y(0)=y(4)=y(5)=y(6) =0
Boundary conditions on z:  z(0) =0
END
```

1.1.5 Features of Linear Programs

Hidden in linear programs are a number of assumptions. The usefulness of this model is directly related to how close reality matches up with these assumptions.

The first two assumptions are due to the linear form of our functions. The contribution to the objective of any decision variable is proportional to the value of the decision variable. Similarly, the contribution of each variable to the left hand side of each constraint is proportional to the value of the variable. This is the *Proportionality Assumption*.

Furthermore, the contribution of a variable to the objective and constraints is independent of the values of the other variables. This is the *Additivity Assumption*.

The next assumption is the *Divisibility Assumption*: is it possible to take any fraction of any variable? A fractional production quantity may be worisome if we are producing a small number of battleships or be innocuous if we are producing millions of paperclips. If the Divisibility Assumption is important and does not hold, then a technique called *integer programming* rather than linear programming is required. This technique takes orders of magnitude more time to find solutions but may be necessary to create realistic solutions.

The final assumption is the *Certainty Assumption*: linear programming allows for no uncertainty about the numbers.

It is very rare that a problem will meet all of the assumptions exactly. That does not negate the usefulness of a model. A model can still give useful managerial insight even if reality differs slightly from the rigorous requirements of the model.

1.1.6 Dedication

Dedication or *cash flow matching* is a technique used to fund known liabilities in the future. The intent is to form a portfolio of assets whose cash inflows will exactly offset the cash outflows of the liabilities. The liabilities will therefore be paid off, as they come due, without the need to sell or buy assets in the future. The portfolio is formed today and then held until all liabilities are paid off. Dedicated portfolios usually only consist of risk-free non-callable bonds since the portfolio future cash inflows need to be known when the portfolio is constructed. This eliminates interest rate risk completely. It is used by some municipalities

and small pension funds. For example, municipalities sometimes want to fund liabilities stemming from bonds they have issued. These pre-refunded municipal bonds can be taken off the books of the municipality. This may allow them to evade restrictive covenants in the bonds that have been pre-refunded and perhaps allow them to issue further debt. It should be noted however that dedicated portfolios cost typically from 3% to 7% more in dollar terms than do "immunized" portfolios that are constructed based on matching present value, duration and convexity of the assets and of the liabilities. Given a stream of cash flows C_t for $t = 1, \ldots, T$, recall that its *present value* is $P = \sum_{t=1}^{T} \frac{C_t}{(1+r_t)^t}$, its *duration* is $D = \frac{1}{P} \sum_{t=1}^{T} \frac{tC_t}{(1+r_t)^t}$ and its *convexity* is $C = \frac{1}{P} \sum_{t=1}^{T} \frac{t(t+1)C_t}{(1+r_t)^{t+2}}$. In these formulas, we will assume that r_t is the risk-free rate in year t. If the portfolio consists only of risk-free bonds, the present value P^* of the portfolio future cash inflows can be computed using the same risk-free rate r_t. Similarly for their duration D^* and convexity C^*. An "immunized" portfolio can be constructed based on matching $P^* = P$, $D^* = D$ and $C^* = C$. Portfolios that are constructed by matching these three factors are immunized against parallel shifts in the yield curve, but there may still be a great deal of exposure and vulnerability to other types of shifts, and they need to be actively managed, which can be costly. By contrast, dedicated portfolios do not need to be managed after they are constructed.

When municipalities use cash flow matching, the standard custom is to call a few investment banks, send them the liability schedule and request bids. The municipality then buys its securities from the bank that offers the lowest price for a successful cash flow match.

A bank receives the following liability schedule:

$Year1$	$Year2$	$Year3$	$Year4$	$Year5$	$Year6$	$Year7$	$Year8$
12,000	18,000	20,000	20,000	16,000	15,000	12,000	10,000

The bonds available for purchase today (Year 0) are given in the next table. All bonds have a face value of \$100. The coupon figure is annual. For example, Bond 5 costs \$98 today, and it pays back \$4 in Year 1, \$4 in Year 2, \$4 in Year 3 and \$104 in Year 4. All these bonds are widely available and can be purchased in any quantities at the stated price.

	Bond1	Bond2	Bond3	Bond4	Bond5	Bond6	Bond7	Bond8	Bond9	Bond10
Price	102	99	101	98	98	104	100	101	102	94
Coupon	5	3.5	5	3.5	4	9	6	8	9	7
Maturity	$Year1$	$Year2$	$Year2$	$Year3$	$Year4$	$Year5$	$Year5$	$Year6$	$Year7$	$Year8$

Formulate and solve a linear program to find the least cost portfolio of bonds to purchase today, to meet the obligations of the municipality over the next eight years. To eliminate the possibility of any reinvestment risk, we assume a 0 % reinvestment rate.

This linear program is available in an Excel spreadsheet called Dedication.xls on the course web page. Using a modeling language, the formulation might look as follows.

```
DATA
LET T=8 be the number of years to plan for.
LET N=10 be the number of bonds available for purchase today.
```

```
L(t) = Liability in year t=1,...,T
P(i) = Price of bond i, i=1,...,N
C(i) = Annual coupon for bond i, i=1,...,N
M(i) = Maturity year of bond i, i=1,...,N
VARIABLES
x(i) = Amount of bond i in the portfolio
z(t) = Surplus in year t-1
OBJECTIVE (Minimize cost)
Min z(1) + SUM(i=1:N) P(i)*x(i)
CONSTRAINTS Year(t=1:T):
SUM(i=1:N | M(i) > t-1) C(i)*x(i) + SUM(i=1:N | M(i) = t) 100*x(i)
-z(t+1) + z(t) = L(t)
END
```

Solving the linear program with SOLVER, we find that we can meet the municipality's liabilities for $93,944 with the following portfolio: 62 Bond1, 125 Bond3, 152 Bond4, 157 Bond5, 123 Bond6, 124 Bond8, 104 Bond9 and 93 Bond10.

1.1.7 Project 1

A municipality sends you the following liability stream (in million dollars):

6/15/03	12/15/03	6/15/04	12/15/04	6/15/05	12/15/05	6/15/06	12/15/06
6	6	9	9	10	10	10	10

6/15/07	12/15/07	6/15/08	12/15/08	6/15/09	12/15/09	6/15/10	12/15/10
8	8	8	8	6	6	5	5

Your job:

- Value the liability using the Treasury curve.
- Identify between 30 and 50 assets that are suitable for a dedicated portfolio (non-callable bonds, treasury bills or notes). Explain why they are suitable. You can find current data on numerous web sites such as bondsonline.com
- Set up a linear program to identify a lowest cost dedicated portfolio of assets (so no short selling) and solve with Excel's solver (or any other linear programming software that you prefer). What is the cost of your portfolio? Discuss the composition of your portfolio. Discuss the assets and the liabilities in light of the Sensitivity Report. What is the term structure of interest rates implied by the shadow prices? Compare with the term structure of treasury rates. (Hint: refer to Section 1.3.2.)
- Set up a linear program to identify a lowest cost portfolio of assets (no short selling) that matches present value, duration and convexity (or a related measure) between the liability stream and the bond portfolio. Solve the linear program with your favorite

software. Discuss the solution. How much would you save by using this immunization strategy instead of dedication? Can you immunize the portfolio against nonparallel shifts of the yield curve? Explain.

- Combine a cash match strategy for the liabilities in the first 3 years and an immunization strategy based on present value, duration and convexity for the liabilities in the last 5 years. Compare the cost of this portfolio with the cost of the two previous portfolios.

- The municipality would like you to make a second bid: what is your lowest cost dedicated portfolio of riskfree bonds if short sales are allowed? Discuss the feasibility of your solution.

This project should be performed individually. It is due on the day of class in Week 3. Turn in a short written report (3 or 4 pages plus appendices if needed).

1.2 The Simplex Method (Optional Material)

In this section, you will learn how to solve linear programs. This will give you insights into what SOLVER and other commercial linear programming software packages actually do. Such an understanding can be useful in several ways. For example, you will be able to identify when a problem has alternate optimal solutions (SOLVER never tells you this: it always give you one optimal solution only). You will also learn about degeneracy in linear programming and how this could lead to a very large number of iterations when trying to solve the problem.

1.2.1 Solving Linear Systems of Equations

To solve linear programs, one solves a sequence of linear systems of equations. Let us briefly review how to solve linear systems of equations.

The Gauss-Jordan elimination procedure is a systematic method for solving systems of linear equations. It works one variable at a time, eliminating it in all rows but one, and then moves on to the next variable.

It is generally true that a system of m linear equations in n variables has either:

(a) no solution,

(b) a unique solution,

(c) infinitely many solutions.

The Gauss-Jordan elimination procedure solves the system of linear equations using two elementary row operations:

- modify some equation by multiplying it by a nonzero scalar (a *scalar* is an actual real number, such as $\frac{1}{2}$ or -2; it cannot be one of the variables in the problem),
- modify some equation by adding to it a scalar multiple of another equation.

The resulting system of m linear equations has the same solution(s) as the original system. If an equation $0 = 0$ is produced, it is discarded and the procedure is continued. If an equation $0 = a$ is produced where a is a nonzero scalar, the procedure is stopped: in this case, the system has no solution. At each step of the procedure, a new variable is made *basic*, that is, it has coefficient 1 in one of the equations and 0 in all the others. The procedure stops when each equation has a basic variable associated with it. Say p equations remain (remember that some of the original m equations may have been discarded). When $n = p$, the system has a unique solution. When $n > p$, then p variables are basic and the remaining $n - p$ are nonbasic. In this case, the system has infinitely many solutions.

Example 1.2.1 (Solving linear equations)

$$\begin{array}{rcrcrcrcl} x_1 & + & 2x_2 & + & x_3 & + & 3x_4 & = & 4 \\ 2x_1 & - & x_2 & + & 3x_3 & + & 2x_4 & = & 3 \\ x_1 & + & x_2 & - & x_3 & & & = & 3 \end{array}$$

In the first step of the procedure, we use the first equation to eliminate x_1 from the other two. Specifically, in order to eliminate x_1 from the second equation, we multiply the first equation by 2 and subtract the result from the second equation. Similarly, to eliminate x_1 from the third equation, we subtract the first equation from the third. Such steps are called *elementary row operations*. We keep the first equation and the modified second and third equations. The resulting equations are

$$\begin{array}{rcrcrcrcr} x_1 & + & 2x_2 & + & x_3 & + & 3x_4 & = & 4 \\ & - & 5x_2 & + & x_3 & - & 4x_4 & = & -5 \\ & - & x_2 & - & 2x_3 & - & 3x_4 & = & -1 \end{array}$$

Note that only one equation was used to eliminate x_1 in all the others. This guarantees that the new system of equations has exactly the same solution(s) as the original one. In the second step of the procedure, we divide the second equation by -5 to make the coefficient of x_2 equal to 1. Then, we use this equation to eliminate x_2 from equations 1 and 3. This yields the following new system of equations.

$$\begin{array}{rcrcrcrcl} x_1 & & & + & \frac{7}{5}x_3 & + & \frac{7}{5}x_4 & = & 2 \\ & x_2 & & - & \frac{1}{5}x_3 & + & \frac{4}{5}x_4 & = & 1 \\ & & & - & \frac{11}{5}x_3 & - & \frac{11}{5}x_4 & = & 0 \end{array}$$

Once again, only one equation was used to eliminate x_2 in all the others and that guarantees that the new system has the same solution(s) as the original one. Finally, in the last step of the procedure, we use equation 3 to eliminate x_3 in equations 1 and 2.

$$\begin{array}{rrrrrrr} x_1 & & & & & = & 2 \\ & x_2 & & + & x_4 & = & 1 \\ & & x_3 & + & x_4 & = & 0 \end{array}$$

The situation where we can express some of the variables (here x_1, x_2 and x_3) in terms of the remaining variables (here x_4) is important when solving linear programs. These variables are said to be *basic* and *nonbasic* respectively. Any choice of the nonbasic variable x_4 yields a solution of the linear system. Therefore the system has infinitely many solutions. This is also the case for the original system of equations because, throughout the Gauss-Jordan procedure, we were careful to keep three equations that have the same solutions as the three original equations.

Exercise 1 Indicate whether the following linear system of equations has 0, 1 or infinitely many solutions.

$$\begin{array}{rrrrrrrrrrr} x_1 & + & 2x_2 & + & 4x_3 & + & x_4 & + & 3x_5 & = & 2 \\ 2x_1 & + & x_2 & + & x_3 & + & 3x_4 & + & x_5 & = & 1 \\ & & 3x_2 & + & 7x_3 & - & x_4 & + & 5x_5 & = & 6 \end{array}$$

Answer: This linear system has no solution.

1.2.2 Solution of Linear Programs by the Simplex Method

For simplicity of exposition, we consider the case where all the constraints are of the form $\leq$ and the right-hand-sides are all nonnegative. We will explain the steps of the simplex method while we progress through an example.

Example 1.2.2 *Solve the linear program*

$$\begin{array}{lrrcl} max & x_1 & +x_2 & & \\ & 2x_1 & +x_2 & \leq & 4 \\ & x_1 & +2x_2 & \leq & 3 \\ & x_1 \geq 0, & x_2 \geq 0 & & \end{array}$$

First, we convert the problem into an equivalent form by adding slack variables $x_3 \geq 0$ and $x_4 \geq 0$.

$$\begin{array}{lrrrrcl} \max & x_1 & +x_2 & & & & \\ & 2x_1 & +x_2 & +x_3 & & = & 4 \\ & x_1 & +2x_2 & & +x_4 & = & 3 \\ & x_1 \geq 0, & x_2 \geq 0 & x_3 \geq 0, & x_4 \geq 0 & & \end{array}$$

Let z denote the objective function value. Here, $z = x_1 + x_2$ or, equivalently,

$$z - x_1 - x_2 = 0.$$

Putting this equation together with the constraints, we get the following system of linear equations.

$$\begin{array}{rrrrrcll} z & -x_1 & -x_2 & & & = & 0 & \text{Row 0} \\ & 2x_1 & +x_2 & +x_3 & & = & 4 & \text{Row 1} \\ & x_1 & +2x_2 & & +x_4 & = & 3 & \text{Row 2} \end{array} \tag{1.1}$$

Our goal is to maximize z, while satisfying these equations and, in addition, $x_1 \geq 0$, $x_2 \geq 0$, $x_3 \geq 0$, $x_4 \geq 0$.

Note that the equations are already in the form that we expect at the last step of the Gauss-Jordan procedure. Namely, the equations are solved in terms of the nonbasic variables x_1, x_2. The variables (other than the special variable z) which appear in only one equation are the *basic variables*. Here the basic variables are x_3 and x_4. A *basic solution* is obtained from the system of equations by setting the nonbasic variables to zero. Here this yields

$$x_1 = x_2 = 0 \quad x_3 = 4 \quad x_4 = 3 \quad z = 0.$$

Is this an optimal solution or can we increase z? (Our goal)

By looking at Row 0 above, we see that we can increase z by increasing x_1 or x_2. This is because these variables have a negative coefficient in Row 0. If all coefficients in Row 0 had been nonnegative, we could have concluded that the current basic solution is optimum, since there would be no way to increase z (remember that all variables x_i must remain ≥ 0). We have just discovered the first rule of the simplex method.

Rule 1 *If all variables have a nonnegative coefficient in Row 0, the current basic solution is optimal.*
Otherwise, pick a variable x_j with a negative coefficient in Row 0.

The variable chosen by Rule 1 is called the *entering* variable. Here let us choose, say, x_1 as our entering variable. It really does not matter which variable we choose as long as it has a negative coefficient in Row 0. The idea is to *pivot* in order to make the nonbasic variable x_1 become a basic variable. In the process, some basic variable will become nonbasic (the *leaving* variable). This *change of basis* is done using the Gauss-Jordan procedure. What is needed next is to choose the *pivot element*. It will be found using Rule 2 of the simplex method. In order to better understand the rationale behing this second rule, let us try both possible pivots and see why only one is acceptable.

First, try the pivot element in Row 1.

$$\begin{array}{rrrrrcll} z & -x_1 & -x_2 & & & = & 0 & \text{Row 0} \\ & \mathbf{2x_1} & +x_2 & +x_3 & & = & 4 & \text{Row 1} \\ & x_1 & +2x_2 & & +x_4 & = & 3 & \text{Row 2} \end{array}$$

This yields

$$\begin{array}{rrrrcll} z & -\frac{1}{2}x_2 & +\frac{1}{3}x_3 & & = & 2 & \text{Row 0} \\ x_1 & +\frac{1}{2}x_2 & +\frac{1}{2}x_3 & & = & 2 & \text{Row 1} \\ & \frac{3}{2}x_2 & -\frac{1}{2}x_3 & +x_4 & = & 1 & \text{Row 2} \end{array}$$

with basic solution $x_2 = x_3 = 0 \quad x_1 = 2 \quad x_4 = 1 \quad z = 2.$

Now, try the pivot element in Row 2.

$$\begin{array}{rrrrcll} z & -x_1 & -x_2 & & = & 0 & \text{Row 0} \\ & 2x_1 & +x_2 & +x_3 & = & 4 & \text{Row 1} \\ & \mathbf{x_1} & +2x_2 & +x_4 & = & 3 & \text{Row 2} \end{array}$$

This yields

$$\begin{array}{rrrrcll} z & & +x_2 & +x_4 & = & 3 & \text{Row 0} \\ & & -3x_2 & +x_3 \quad -2x_4 & = & -2 & \text{Row 1} \\ & x_1 & +2x_2 & +x_4 & = & 3 & \text{Row 2} \end{array}$$

with basic solution $x_2 = x_4 = 0 \quad x_1 = 3 \quad x_3 = -2 \quad z = 3.$

Which pivot should we choose? The first one, of course, since the second yields an *infeasible* basic solution! Indeed, remember that we must keep all variables ≥ 0. With the second pivot, we get $x_3 = -2$ which is infeasible. How could we have known this ahead of time, before actually performing the pivots? The answer is, by comparing the ratios $\frac{\text{Right Hand Side}}{\text{Entering Variable Coefficient}}$ in Rows 1 and 2 of (1.1). Here we get $\frac{4}{2}$ in Row 1 and $\frac{3}{1}$ in Row 2. If you pivot in a row with *minimum* ratio, you will end up with a feasible basic solution (i.e. you will not introduce negative entries in the Right Hand Side), whereas if you pivot in a row with a ratio which is not minimum you will always end up with an infeasible basic solution. Just simple algebra! A negative pivot element would not be good either, for the same reason. We have just discovered the second rule of the simplex method.

Rule 2 *For each Row i, $i \geq 1$, where there is a strictly positive "entering variable coefficient", compute the ratio of the Right Hand Side to the "entering variable coefficient". Choose the pivot row as being the one with MINIMUM ratio.*

Once you have identified the pivot element by Rule 2, you perform a Gauss-Jordan pivot. This gives you a new basic solution. Is it an optimal solution? This question is addressed by Rule 1, so we have closed the loop. The simplex method iterates between Rules 1, 2 and pivoting until Rule 1 guarantees that the current basic solution is optimal. That's all there is to the simplex method.

After our first pivot, we obtained the following system of equations.

$$\begin{array}{rrrrcll} z & & -\frac{1}{2}x_2 & +\frac{1}{3}x_3 & = & 2 & \text{Row 0} \\ & x_1 & +\frac{1}{2}x_2 & +\frac{1}{2}x_3 & = & 2 & \text{Row 1} \\ & & \frac{3}{2}x_2 & -\frac{1}{2}x_3 \quad +x_4 & = & 1 & \text{Row 2} \end{array}$$

with basic solution $x_2 = x_3 = 0 \quad x_1 = 2 \quad x_4 = 1 \quad z = 2.$

Is this solution optimal? No, Rule 1 tells us to choose x_2 as entering variable. Where should we pivot? Rule 2 tells us to pivot in Row 2, since the ratios are $\frac{2}{1/2} = 4$ for Row 1,

and $\frac{1}{3/2} = \frac{2}{3}$ for Row 2, and the minimum occurs in Row 2. So we pivot on $\frac{3}{2}x_2$ in the above system of equations. This yields

$$\begin{array}{rrrrcll} z & & +\frac{1}{3}x_3 & +\frac{1}{3}x_4 & = & \frac{7}{3} & \text{Row 0} \\ x_1 & & +\frac{2}{3}x_3 & -\frac{1}{3}x_4 & = & \frac{5}{3} & \text{Row 1} \\ & x_2 & -\frac{1}{3}x_3 & +\frac{2}{3}x_4 & = & \frac{2}{3} & \text{Row 2} \end{array}$$

with basic solution $x_3 = x_4 = 0 \quad x_1 = \frac{5}{3} \quad x_2 = \frac{2}{3} \quad z = \frac{7}{3}$.

Now Rule 1 tells us that this basic solution is optimal, since there are no more negative entries in Row 0.

All the above computations can be represented very compactly in *tableau form*.

z	x_1	x_2	x_3	x_4	RHS		Basic solution
1	−1	−1	0	0	0	basic	$x_3 = 4 \quad x_4 = 3$
0	2	1	1	0	4	nonbasic	$x_1 = x_2 = 0$
0	1	2	0	1	3	$z = 0$	
1	0	$-\frac{1}{2}$	$\frac{1}{2}$	0	2	basic	$x_1 = 2 \quad x_4 = 1$
0	1	$\frac{1}{2}$	$\frac{1}{2}$	0	2	nonbasic	$x_2 = x_3 = 0$
0	0	$\frac{3}{2}$	$-\frac{1}{2}$	1	1	$z = 2$	
1	0	0	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{7}{3}$	basic	$x_1 = \frac{5}{3} \quad x_2 = \frac{2}{3}$
0	1	0	$\frac{2}{3}$	$-\frac{1}{3}$	$\frac{5}{3}$	nonbasic	$x_3 = x_4 = 0$
0	0	1	$-\frac{1}{3}$	$\frac{2}{3}$	$\frac{2}{3}$	$z = \frac{7}{3}$	

Graphical Interpretation

Since the above example has only two variables, it is interesting to interpret the steps of the simplex method graphically. See Figure 1.1. The simplex method starts in the corner point $(x_1 = 0, x_2 = 0)$ with $z = 0$. Then it discovers that z can increase by increasing, say, x_1. Since we keep $x_2 = 0$, this means we move along the x_1 axis. How far can we go? Only until we hit a constraint: if we went any further, the solution would become infeasible. That's exactly what Rule 2 of the simplex method does: the minimum ratio rule identifies the first constraint that will be encountered. And when the constraint is reached, its slack x_3 becomes zero. So, after the first pivot, we are at the point $(x_1 = 2, x_2 = 0)$. Rule 1 discovers that z can be increased by increasing x_2 while keeping $x_3 = 0$. This means that we move along the boundary of the feasible region $2x_1 + x_2 = 4$ until we reach another constraint! After pivoting, we reach the optimal point $(x_1 = \frac{5}{3}, x_2 = \frac{2}{3})$.

Exercise 2 Solve the following linear program by the simplex method.

$$\begin{array}{lrrrcl} \max & 4x_1 & +x_2 & -x_3 & & \\ & x_1 & & +3x_3 & \leq & 6 \\ & 3x_1 & +x_2 & +3x_3 & \leq & 9 \\ & x_1 \geq 0, & x_2 \geq 0 & x_3 \geq 0 & & \end{array}$$

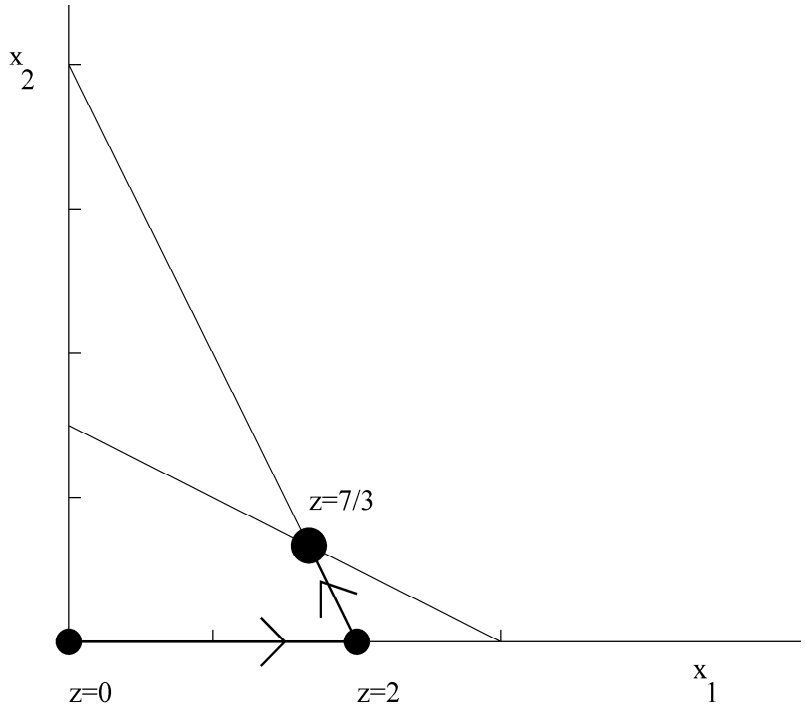

Figure 1.1: Graphical Interpretation

Answer:

z	x_1	x_2	x_3	s_1	s_2	RHS
1	-4	-1	1	0	0	0
0	1	0	3	1	0	6
0	3	1	3	0	1	9
1	0	$\frac{1}{3}$	5	0	$\frac{4}{3}$	12
0	0	$\frac{-1}{3}$	2	1	$\frac{-1}{3}$	3
0	1	$\frac{1}{3}$	1	0	$\frac{1}{3}$	3

The optimal solution is $x_1 = 3$, $x_2 = x_3 = 0$.

1.2.3 Alternate Optimal Solutions, Degeneracy, Unboudedness, Infeasibility

Alternate Optimal Solutions

Let us solve a small variation of the earlier example, with the same constraints but a slightly different objective:

$$\begin{array}{lrrcl} \max & x_1 & +\frac{1}{2}x_2 & & \\ & 2x_1 & +x_2 & \leq & 4 \\ & x_1 & +2x_2 & \leq & 3 \\ & x_1 \geq 0, & x_2 \geq 0 & & \end{array}$$

As before, we add slacks x_3 and x_4, and we solve by the simplex method, using tableau representation.

z	x_1	x_2	x_3	x_4	RHS		Basic solution
1	-1	$-\frac{1}{2}$	0	0	0	basic	$x_3 = 4 \quad x_4 = 3$
0	2	1	1	0	4	nonbasic	$x_1 = x_2 = 0$
0	1	2	0	1	3	$z = 0$	
1	0	0	$\frac{1}{2}$	0	2	basic	$x_1 = 2 \quad x_4 = 1$
0	1	$\frac{1}{2}$	$\frac{1}{2}$	0	2	nonbasic	$x_2 = x_3 = 0$
0	0	$\frac{3}{2}$	$-\frac{1}{2}$	1	1	$z = 2$	

Now Rule 1 shows that this is an optimal solution. Interestingly, the coefficient of the nonbasic variable x_2 in Row 0 happens to be equal to 0. Going back to the rationale that allowed us to derive Rule 1, we observe that, if we increase x_2 (from its current value of 0), this will not effect the value of z. Increasing x_2 produces changes in the other variables, of course, through the equations in Rows 1 and 2. In fact, we can use Rule 2 and pivot to get a different basic solution with the same objective value $z = 2$.

z	x_1	x_2	x_3	x_4	RHS		Basic solution
1	0	0	$\frac{1}{3}$	0	2	basic	$x_1 = \frac{5}{3} \quad x_2 = \frac{2}{3}$
0	1	0	$\frac{2}{3}$	$-\frac{1}{3}$	$\frac{5}{3}$	nonbasic	$x_3 = x_4 = 0$
0	0	1	$-\frac{1}{3}$	$\frac{2}{3}$	$\frac{2}{3}$	$z = 2$	

Note that the coefficient of the nonbasic variable x_4 in Row 0 is equal to 0. Using x_4 as entering variable and pivoting, we would recover the previous solution!

Degeneracy

Example 1.2.3

$$\begin{array}{lrrcl} max & 2x_1 & +x_2 & & \\ & 3x_1 & +x_2 & \leq & 6 \\ & x_1 & -x_2 & \leq & 2 \\ & & x_2 & \leq & 3 \\ & x_1 \geq 0, & x_2 \geq 0 & & \end{array}$$

Let us solve this problem using the –by now familiar– simplex method. In the initial tableau, we can choose x_1 as the entering variable (Rule 1) and Row 2 as the pivot row (the minimum ratio in Rule 2 is a tie, and ties are broken arbitrarily). We pivot and this yields the second tableau below.

z	x_1	x_2	x_3	x_4	x_5	RHS		Basic solution
1	-2	-1	0	0	0	0	basic	$x_3 = 6 \quad x_4 = 2 \quad x_5 = 3$
0	3	1	1	0	0	6	nonbasic	$x_1 = x_2 = 0$
0	1	-1	0	1	0	2	$z = 0$	
0	0	1	0	0	1	3		
1	0	-3	0	2	0	4	basic	$x_1 = 2 \quad x_3 = 0 \quad x_5 = 3$
0	0	4	1	-3	0	0	nonbasic	$x_2 = x_4 = 0$
0	1	-1	0	1	0	2	$z = 4$	
0	0	1	0	0	1	3		

Note that this basic solution has a basic variable (namely x_3) which is equal to zero. When this occurs, we say that the basic solution is *degenerate*. Should this be of concern? Let us continue the steps of the simplex method. Rule 1 indicates that x_2 is the entering variable. Now let us apply Rule 2. The ratios to consider are $\frac{0}{4}$ in Row 1 and $\frac{3}{1}$ in Row 3. The minimum ratio occurs in Row 1, so let us perform the corresponding pivot.

z	x_1	x_2	x_3	x_4	x_5	RHS		Basic solution			
1	0	0	$\frac{3}{4}$	$-\frac{1}{4}$	0	4		basic	$x_1 = 2$	$x_2 = 0$	$x_5 = 3$
0	0	1	$\frac{1}{4}$	$-\frac{3}{4}$	0	0		nonbasic	$x_3 = x_4 = 0$		
0	1	0	$\frac{1}{4}$	$\frac{1}{4}$	0	2		$z = 4$			
0	0	0	$-\frac{1}{4}$	$\frac{3}{4}$	1	3					

We get exactly the same solution! The only difference is that we have interchanged the names of a nonbasic variable with that of a degenerate basic variable (x_2 and x_3). Rule 1 tells us the solution is not optimal, so let us continue the steps of the simplex method. Variable x_4 is the entering variable and the last row wins the minimum ratio test. After pivoting, we get the tableau:

z	x_1	x_2	x_3	x_4	x_5	RHS		Basic solution			
1	0	0	$\frac{2}{3}$	0	$\frac{1}{3}$	5		basic	$x_1 = 1$	$x_2 = 3$	$x_4 = 4$
0	0	1	0	0	1	3		nonbasic	$x_3 = x_5 = 0$		
0	1	0	$\frac{1}{3}$	0	$-\frac{1}{3}$	1		$z = 5$			
0	0	0	$-\frac{1}{3}$	1	$\frac{4}{3}$	4					

By Rule 1, this is the optimal solution. So, after all, degeneracy did not prevent the simplex method to find the optimal solution in this example. It just slowed things down a little. Unfortunately, on other examples, degeneracy may lead to *cycling*, i.e. a sequence of pivots that goes through the same tableaus and repeats itself indefinitely. In theory, cycling can be avoided by choosing the entering variable with smallest index in Rule 1, among all those with a negative coefficient in Row 0, and by breaking ties in the minimum ratio test by choosing the leaving variable with smallest index (this is known as Bland's rule). This rule, although it guaranties that cycling will never occur, turns out to be somewhat inefficient. Actually, in commercial codes, no effort is made to avoid cycling. This may come as a surprise, since degeneracy is a frequent occurence. But there are two reasons for this:

- Although degeneracy is frequent, cycling is extremely rare.
- The precision of computer arithmetic takes care of cycling by itself: round off errors accumulate and eventually gets the method out of cycling.

Our example of degeneracy is a 2-variable problem, so you might want to draw the constraint set in the plane and interpret degeneracy graphically.

Unbounded Optimum

Example 1.2.4

$$\begin{array}{lrrcl} max & 2x_1 & +x_2 & & \\ & -x_1 & +x_2 & \leq & 1 \\ & x_1 & -2x_2 & \leq & 2 \\ & x_1 \geq 0, & x_2 \geq 0 & & \end{array}$$

Solving by the simplex method, we get:

z	x_1	x_2	x_3	x_4	RHS		Basic solution	
1	-2	-1	0	0	0	basic	$x_3 = 1$	$x_4 = 2$
0	-1	1	1	0	1	nonbasic	$x_1 = x_2 = 0$	
0	1	-2	0	1	2	$z = 0$		
1	0	-5	0	2	4	basic	$x_1 = 2$	$x_3 = 3$
0	0	-1	1	1	3	nonbasic	$x_2 = x_4 = 0$	
0	1	-2	0	1	2	$z = 4$		

At this stage, Rule 1 chooses x_2 as the entering variable, but there is no ratio to compute, since there is no positive entry in the column of x_2. As we start increasing x_2, the value of z increases (from Row 0) and the values of the basic variables increase as well (from Rows 1 and 2). There is nothing to stop them going off to infinity. So the problem is unbounded.

Interpret the steps of the simplex method graphically for this example.

Infeasible Linear Programs

It is easy to construct constraints that have no solution. The simplex method is able to identify such cases. We do not discuss it here. The interested reader is referred to a textbook on linear programming. There are many such books. An excellent one is

V. Chvátal, *Linear Programming*, Freeman (1983).

Properties of Linear Programs

There are three possible outcomes for a linear program: it is infeasible, it has an unbounded optimum or it has an optimal solution.

If there is an optimal solution, there is a *basic* optimal solution. Remember that the number of basic variables in a basic solution is equal to the number of constraints of the problem, say m. So, even if the total number of variables, say n, is greater than m, at most m of these variables can have a positive value in an optimal basic solution.

Exercise 3 The following tableaus were obtained in the course of solving linear programs with 2 nonnegative variables x_1 and x_2 and 2 inequality constraints (the objective function z is maximized). Slack variables s_1 and s_2 were added. In each case, indicate whether the linear program

(i) is unbounded

(ii) has a unique optimum solution

(iii) has an alternate optimum solution

(iv) is degenerate (in this case, indicate whether any of the above holds).

(a)

z	x_1	x_2	s_1	s_2	RHS
1	0	3	2	0	20
0	1	−2	−1	0	4
0	0	−1	0	1	2

(b)

z	x_1	x_2	s_1	s_2	RHS
1	0	−1	0	2	20
0	0	0	1	−2	5
0	1	−2	0	3	6

(c)

z	x_1	x_2	s_1	s_2	RHS
1	2	0	0	1	8
0	3	1	0	−2	4
0	−2	0	1	1	0

(d)

z	x_1	x_2	s_1	s_2	RHS
1	0	0	2	0	5
0	0	−1	1	1	4
0	1	1	−1	0	4

Answer:
(a) Unique optimum solution.
(b) Unbounded linear program.
(c) Unique degenerate optimum solution.
(d) Alternate optimum solution.

Exercise 4 Suppose the following tableau was obtained in the course of solving a linear program with nonnegative variables x_1, x_2, x_3 and two inequalities. The objective function is maximized and slack variables s_1 and s_2 were added.

z	x_1	x_2	x_3	s_1	s_2	RHS
1	0	a	b	0	4	82
0	0	−2	2	1	3	c
0	1	−1	3	0	−5	3

Give conditions on a, b and c that are required for the following statements to be true:

(i) The current basic solution is a feasible basic solution.

Assume that the condition found in (i) holds in the rest of the exercise.

(ii) The current basic solution is optimal.

(iii) The linear program is unbounded (for this question, assume that $b > 0$).

(iv) The currrent basic solution is optimal and there are alternate optimal solutions (for this question, assume $a > 0$).

Answer:
(i) $c \geq 0$.
(ii) $a \geq 0$ and $b \geq 0$.
(iii) $a < 0$.
(iv) $b = 0$.

Exercise 5 A plant can manufacture five products P_1, P_2, P_3, P_4 and P_5. The plant consists of two work areas: the job shop area A_1 and the assembly area A_2. The time required to process one unit of product P_j in work area A_i is p_{ij} (in hours), for $i = 1, 2$ and $j = 1, \ldots, 5$. The weekly capacity of work area A_i is C_i (in hours). The company can sell all it produces of product P_j at a profit of s_j, for $i = 1, \ldots, 5$.

The plant manager thought of writing a linear program to maximize profits, but never actually did for the following reason: From past experience, he observed that the plant operates best when at most two products are manufactured at a time. He believes that if he uses linear programming, the optimal solution will consist of producing all five products. Do you agree with him? Explain, based on your knowledge of linear programming.

Answer: The linear program has two constraints (one for each of the work areas). Therefore, at most two variables are positive in a basic solution. In particular, this is the case for an optimal solution. So the plant manager is mistaken in his beliefs about linear programming.

1.2.4 Alternative to the Simplex Method

Performing a pivot of the simplex method is extremely fast on today's computers, even for problems with thousands of variables and hundreds of constraints. This explains the success of the simplex method. However, for large problems, the number of iterations can be enormous. What do we mean by a "large" linear program? We mean a problem with several thousands variables and constraints, say 5,000 constraints and 100,000 variables or more. Such models are not uncommon in financial applications and can often be handled by the simplex method. In recent years, another method has emerged as an alternative. It is known under the name of *barrier method* or *interior point method*. It uses a totally different strategy to reach the optimum, following a path in the interior of the feasible region. Each iteration is fairly expensive, but the number of iterations needed does not depend much on the size of the problem. As a result, barrier methods can be faster than the simplex method for large scale problems (thousands of constraints). Most state-of-the-art linear programming packages (Cplex, Xpress, OSL, etc) give you the option to solve your linear programs by either method.

1.3 Sensitivity Analysis for Linear Programming

Finding the optimal solution to a linear programming model is important, but it is not the only information available. There is a tremendous amount of *sensitivity information*, or information about what happens when data values are changed.

Recall that in order to formulate a problem as a linear program, we had to invoke a *certainty assumption*: we had to know what value the data took on, and we made decisions based on that data. Often this assumption is somewhat dubious: the data might be unknown, or guessed at, or otherwise inaccurate. How can we determine the effect on the optimal decisions if the values change? Clearly some numbers in the data are more important than others. Can we find the "important" numbers? Can we determine the effect of misestimation?

Linear programming offers extensive capabilities for addressing these questions. We give examples of how to interpret the SOLVER output. To access the information, simply ask for the sensitivity report after optimizing. Rather than simply giving rules for reading the reports, we show how to answer a set of questions from the output.

1.3.1 Short Term Financing

The Solver sensitivity report looks as follows.

Adjustable Cells

Cell	Name	Final Value	Reduced Cost	Objective Coefficient	Allowable Increase	Allowable Decrease
$B$2	$x1$	0	-0.0032	0	0.0032	$1E+30$
$B$3	$x2$	50.98	0	0	0.0032	0
$B$4	$x3$	0	-0.0071	0	0.0071	$1E+30$
$B$5	$x4$	0	-0.0032	0	0.0032	$1E+30$
$B$6	$x5$	0	0	0	0	$1E+30$
$C$2	$y1$	150	0	0	0.0040	0.0032
$C$3	$y2$	49.02	0	0	0	0.0032
$C$4	$y3$	203.43	0	0	0.0071	0
$D$2	$z1$	0	-0.0040	0	0.0040	$1E+30$
$D$3	$z2$	0	-0.0071	0	0.0071	$1E+30$
$D$4	$z3$	351.94	0	0	0.0039	0.0032
$D$5	$z4$	0	-0.0039	0	0.0039	$1E+30$
$D$6	$z5$	0	-0.007	0	0.007	$1E+30$
$E$2	v	92.50	0	1	$1E+30$	1

Constraints

Cell	Name	Final Value	Shadow Price	Constraint R.H.Side	Allowable Increase	Allowable Decrease
$B$10	january	150	−1.0373	150	89.17	150
$B$11	february	100	−1.030	100	49.020	50.980
$B$12	march	−200	−1.020	−200	90.683	203.434
$B$13	april	200	−1.017	200	90.955	204.044
$B$14	may	−50	−1.010	−50	50	52
$B$15	june	−300	−1	−300	92.497	$1E+30$
$B$16	$x1$	0	0	100	$1E+30$	100
$B$17	$x2$	50.98	0	100	$1E+30$	49.020
$B$18	$x3$	0	0	100	$1E+30$	100
$B$19	$x4$	0	0	100	$1E+30$	100
$B$20	$x5$	0	0	100	$1E+30$	100

The key columns for sensitivity analysis are the `Reduced Cost` and `Shadow Price` columns in SOLVER. The *shadow price* u of a constraint C has the following interpretation:

> If the right hand side of the constraint C increases by an amount Δ, the optimal objective value changes by $u\Delta$.

For a linear program, the shadow price u is an exact figure, as long as the amount of change Δ is within the allowable range given in the last two columns of the SOLVER output. When the change Δ falls outside this range, the shadow price u cannot be used. When this occurs, one has to resolve the linear program using the new data.

For example, assume that Net Cash Flow in january were -200 (instead of - 150). By how much would the company's wealth decrease at the end of june?

The answer is in the shadow price of the january constraint, $u = -1.0373$. The RHS of the january constraint would go from 150 to 200, an increase of $\Delta = 50$, which is within the allowable increase (89.17). So the company's wealth would decrease by 1.0373 * 50,000 = $ 51,865.

Now assume that Net Cash Flow in march were 250 (instead of 200). By how much would the company's wealth increase at the end of june?

Again, the change $\Delta = -50$ is within the allowable decrease (203.434), so we can use the shadow price $u = -1.02$ to calculate the change in objective value. The increase is (-1.02) * (-50) = $51,000.

Assume that the credit limit were increased from 100 to 200. By how much would the company's wealth increase at the end of june?

The change $\Delta = 100$ is within the allowable increase ($+\infty$) and the shadow price is $u = 0$. So there is no effect on the company's wealth in june.

Assume that the negative Net Cash Flow in january is due to the purchase of a machine worth $150,000. The vendor allows the payment to be made in june at an interest rate of 3% for the 5-month period. Would the company's wealth increase or decrease by using this option? What if the interest rate for the 5-month period were 4%?

The shadow price of the january constraint is -1.0373. This means that reducing cash requirements in january by \$1 increase wealth by \$1.0373. In other words, the break even interest rate for the 5-month period is 3.73%. So, if the vendor charges 3%, we should accept, but if he charges 4% we should not. Note that the analysis is valid since the amount $\Delta = -150$ is within the allowable decrease.

Now, let us consider the reduced costs. The basic variables always have a zero reduced cost. The nonbasic variables (which by definition take the value 0) frequently have a nonzero reduced cost. The usual convention is that, for a maximization problem, the reduced costs are nonnegative. SOLVER has the opposite convention, but the magnitude of the numbers are correct! There are two useful interpretations of the reduced cost c, for a nonbasic variable x.

First, assume that x is set to a positive value Δ instead of its optimal value 0. Then, the objective value is decreased by $c\Delta$. For example, what would be the effect of financing part of the january cash needs through the line of credit? The answer is in the reduced cost of variable x_1. The effect would be a decrease in the company's wealth v by \$0.0032 per dollar.

The second interpretation of c is that it is the minimum amount by which the objective coefficient of x must be increased in order for the variable x to become positive in an optimal solution. For example, consider the variable x_1 again. Its value is zero in the current optimal solution, with objective function v. Hovewer, if we changed the objective to $v + 0.0032x_1$, it would now be optimal to use the line of credit in january. In other words, in this example, the reduced cost on x_1 can be viewed as the minimum rebate that the bank would have to offer (payable in june) to make it attractive to use the line of credit in january.

1.3.2 Dedication

We end this section with the sensitivity report of the dedication problem formulated in Section 1.1.6.

Adjustable Cells

Cell	Name	Final Value	Reduced Cost	Objective Coefficient	Allowable Increase	Allowable Decrease
$B$5	$x1$	62.13612744	0	102	3	5.590909091
$B$6	$x2$	0	0.830612245	99	$1E+30$	0.830612245
$B$7	$x3$	125.2429338	0	101	0.842650104	3.311081442
$B$8	$x4$	151.5050805	0	98	3.37414966	4.712358277
$B$9	$x5$	156.8077583	0	98	4.917243419	17.2316607
$B$10	$x6$	123.0800686	0	104	9.035524153	3.74817022
$B$11	$x7$	0	8.786840002	100	$1E+30$	8.786840002
$B$12	$x8$	124.1572748	0	101	3.988878399	8.655456271
$B$13	$x9$	104.0898568	0	102	9.456887408	0.860545483
$B$14	$x10$	93.45794393	0	94	0.900020046	$1E+30$
$H$4	$z0$	0	0.028571429	1	$1E+30$	0.028571429
$H$5	$z1$	0	0.055782313	0	$1E+30$	0.055782313
$H$6	$z2$	0	0.03260048	0	$1E+30$	0.03260048
$H$7	$z3$	0	0.047281187	0	$1E+30$	0.047281187
$H$8	$z4$	0	0.179369792	0	$1E+30$	0.179369792
$H$9	$z5$	0	0.036934059	0	$1E+30$	0.036934059
$H$10	$z6$	0	0.086760435	0	$1E+30$	0.086760435
$H$11	$z7$	0	0.008411402	0	$1E+30$	0.008411402

Constraints

Cell	Name	Final Value	Shadow Price	Constraint R.H.Side	Allowable Increase	Allowable Decrease
$B$19	$year1$	12000	0.971428571	12000	$1E+30$	6524.293381
$B$20	$year2$	18000	0.915646259	18000	137010.161	13150.50805
$B$21	$year3$	20000	0.883045779	20000	202579.3095	15680.77583
$B$22	$year4$	20000	0.835764592	20000	184347.1716	16308.00686
$B$23	$year5$	16000	0.6563948	16000	89305.96314	13415.72748
$B$24	$year6$	15000	0.619460741	15000	108506.7452	13408.98568
$B$25	$year7$	12000	0.532700306	12000	105130.9798	11345.79439
$B$26	$year8$	10000	0.524288903	10000	144630.1908	10000

Exercise 6

- *Interpret the shadow price in year t ($t = 1, \ldots, 8$)*
- *Interpret the reduced cost of bond i ($i = 1, \ldots, 10$)*
- *Interpret the reduced cost of each surplus variable z_t ($t = 0, \ldots, 7$)*

Answers:

The shadow price in year t is the cost of the bond portfolio that can be attributed to a dollar of liability in year t. For example, each dollar of liability in year 3 is responsible for \$ 0.883 in the cost of the bond portfolio. Note that, by setting the shadow price in year t equal to $\frac{1}{(1+r_t)^t}$, we get a term structure of interest rates. Here $r_3 = 0.0423$. How does this compare with the term structure of treasury rates?

The reduced cost of bond i indicates by how much bond i is overpriced for inclusion in the optimal portfolio. For example, bond 2 would have to be \$ 0.83 lower, at \$ 98.17, for inclusion in the optimal portfolio. Note that bond 7 appears to be completely mispriced at \$ 100. A more realistic price would be just above \$ 91. By checking the reduced costs, one may sometimes spot errors in the data!

The reduced cost of the surplus variable z_t indicates what the interest rate on cash reinvested in year t would have to be in order to keep excess cash in year t.

1.4 Five Other Linear Programming Models

Linear programming models are found in almost every field of business (and beyond!). The next sections go through a number of examples in finance, showing how to model them with the appropriate choice of decision variables, objective, and constraints. In all cases, we will describe the problem and give a model.

1.4.1 Options Problem

You have \$20,000 to invest. Stock XYZ sells at \$20 per share today. A European call option to buy a share of stock XYZ at \$15 exactly six months from today sells for \$10. You can also raise additional funds which can be immediately invested, if desired, by selling call options with the above characteristics. In addition, a 6-month riskless zero-coupon bond with \$100 face value sells for \$90. You have decided to limit the number of call options that you buy or sell to at most 5,000.

You consider three scenarios for the price of stock XYZ six months from today: the price will be the same as today, the price will go up to \$40, or drop to \$12. Your best estimate is that each of these scenarios is equaly likely. Formulate and solve a linear program to determine the portfolio of stocks, bonds, and options that maximize expected profit.

Model

First, we define the decision variables.

B = number of bonds purchased,
S = number of shares of stock XYZ purchased,
C = number of call options purchased (if > 0) or sold (if < 0).

The expected profits (per unit of investment) are computed as follows.

Bonds: 10
Stock XYZ: $\frac{1}{3}(20 + 0 - 8) = 4$
Call Option: $\frac{1}{3}(15 - 5 - 10) = 0$

Therefore, we get the following linear programming formulation.

```
   max 10 B + 4 S
90 B + 20 S + 10 C ≤ 20000 (budget constraint)
C ≤ 5000 (limit on number of call options purchased)
C ≥ -5000 (limit on number of call options sold)
B ≥ 0, S ≥ 0 (nonnegativity).
```

Solving (using SOLVER, say), we get the optimal solution $B = 0$, $S = 3500$, $C =$ -5000 with an expected profit of \$14,000.

Note that, with this portfolio, the profit is not positive under all scenarios. In particular, if the price of stock XYZ goes to \$40, a loss of \$5000 will be incured. Suppose that the investor wants a profit of at least \$2000 in any of the three scenarios. Write a linear program that will maximize the investor's expected profit under this additional constraint.

This can be done by introducing three additional variables.

P_i = profit in scenario i

The formulation is now the following.

$$
\begin{array}{lllllll}
\max & \frac{1}{3}W_1 & +\frac{1}{3}W_2 & +\frac{1}{3}W_3 & & \\
 & 90B & +20S & +10C & \leq & 20000 \\
 & 10B & +20S & +15C & = & P_1 \\
 & 10B & & -5C & = & P_2 \\
 & 10B & -8S & -10C & = & P_3 \\
 & P_1 & & & \geq & 2000 \\
 & P_2 & & & \geq & 2000 \\
 & P_3 & & & \geq & 2000 \\
 & C & & & \leq & 5000 \\
 & C & & & \geq & -5000 \\
 & B \geq 0, & S \geq 0. & & &
\end{array}
$$

Solve this linear program with SOLVER to find out the expected profit. How does it compare with the earlier figure of \$14,000?

Answer: The optimum solution is to buy 2,800 shares of XYZ and sell 3,600 call options. The resulting expected worth in six months will be \$31,200. Therefore, the expected profit is \$11,200 (=\$31,200 - 20,000).

Riskless profit is defined as the smallest possible profit that a portfolio will earn, no matter which of the three scenarios occurs. What is the portfolio that maximizes riskless profit?

To solve this question, we can use a slight modification of the previous model, by introducing one more variable.

Z = riskless profit.

Here is the formulation.

$$
\begin{array}{lllllll}
\max & Z & & & & & \\
 & 90B & +20S & +10C & \leq & 20000 \\
 & 10B & +20S & +15C & = & P_1 \\
 & 10B & & -5C & = & P_2 \\
 & 10B & -8S & -10C & = & P_3 \\
 & P_1 & & & \geq & Z \\
 & P_2 & & & \geq & Z \\
 & P_3 & & & \geq & Z \\
 & C & & & \leq & 5000 \\
 & C & & & \geq & -5000 \\
 & B \geq 0, & S \geq 0. & & & &
\end{array}
$$

The result is (obtained using SOLVER) a riskless profit of \$7272. This is obtained by buying 2,273 shares of XYZ and selling 2,545 call options. The resulting expected profit is \$9,091 in this case.

1.4.2 Portfolio Optimization

Problem Definition.

The optimality of a portfolio depends heavily on the model used for defining risk and other aspects of financial instruments. Here is a particularly simple model that is amenable to linear programming techniques.

Consider a mortgage team with \$100,000,000 to finance various investments. There are five categories of loans, each with an associated return and risk (1-10, 1 best):

Loan/investment	Return (%)	Risk
First Mortgages	9	3
Second Mortgages	12	6
Personal Loans	15	8
Commercial Loans	8	2
Government Securities	6	1

Any uninvested money goes into a savings account with no risk and 3% return. The goal for the mortgage team is to allocate the money to the categories so as to:

(a) Maximize the average return per dollar,

(b) Have an average risk of no more than 5 (all averages and fractions taken over the invested money (not over the saving account)),

(c) Invest at least 20% in commercial loans,

(d) The amount in second mortgages and personal loans combined should be no higher than the amount in first mortgages.

Model

Let the investments be numbered $1 \ldots 5$, and let x_i be the amount invested in investment i. Let x_s be the amount in the savings account. The objective is to maximize

$$9x_1 + 12x_2 + 15x_3 + 8x_4 + 6x_5 + 3x_s$$

subject to

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_s = 100,000,000.$$

Now, let's look at the average risk. Since we want to take the average over only the invested amount, a direct translation of this constraint is

$$\frac{3x_1 + 6x_2 + 8x_3 + 2x_4 + x_5}{x_1 + x_2 + x_3 + x_4 + x_5} \leq 5$$

This constraint is not linear, but we can cross multiply, and simplify to get the equivalent linear constraint:

$$-2x_1 + x_2 + 3x_3 - 3x_4 - 4x_5 \leq 0$$

Similarly we need

$$x_4 \geq 0.2(x_1 + x_2 + x_3 + x_4 + x_5)$$

or

$$-0.2x_1 - 0.2x_2 - 0.2x_3 + 0.8x_4 - 0.2x_5 \geq 0$$

The final constraint is

$$x_2 + x_3 - x_1 \leq 0$$

Together with nonnegativity, this gives the entire formulation.

Solving it, we find that we can achieve a return of 11.2% by investing 40% of the assets in First Mortgages, 40% in Personal Loans and 20% in Commercial Loans.

Discussion

Linear programming models provide great modeling power with a great limit: the handling of risk must be done in a linear fashion (like our Risk factors here). Other classical models look at the co-variance of returns between investments, a fundamentally nonlinear effect. This gives rise to nonlinear models like those that try to minimize variance subject to return requirements. We will see such nonlinear programming models in Chapter 2.

1.4.3 Who is bankrupt?

Albert, Bill, Charles, David, and Edward have gotten into a bind. After a series of financial transactions, they have ended up each owing some of the others huge amounts of money. In fact, near as the lawyers can make out, the debts are as follows

Debtor	Creditor	Amount ($millions)
A	E	10
A	C	3
B	A	5
C	B	6
C	D	4
D	A	4
E	C	7
E	D	3

The question is, who is bankrupt? We will say that a person i is bankrupt if there is no possible transfer of funds among the people such that i completely pays off his obligations. For instance, Albert is bankrupt since he owes 13, and is only owed 9. Edward is bankrupt as well since he owes 10 and, although he is owed 10, this debt is owed to him by Albert who can pay at most 9. Formulate the problem of determining whether Bill is bankrupt, as a linear program. Then modify your formulation to determine if each of the others is bankrupt. *This example may look contrived, but it is inspired by a solution to the debts resulting from a crash of Kuwait's al-Mankh stock market.*

Answer:

Here is a model to decide whether Bill is bankrupt.

```
VARIABLES
AE = Transfer from Albert to Edward
AC = Transfer from Albert to Charles
BA = Transfer from Bill to Albert
CB = Transfer from Charles to Bill
CD = Transfer from Charles to David
DA = Transfer from David to Albert
EC = Transfer from Edward to Charles
ED = Transfer from Edward to David
OBJECTIVE (minimize the debt of Bill)
Min BA
CONSTRAINTS
AlbertTransfers:  BA + DA - AC - AE = 0
BillTransfers:  BA - CB = 0
CharlesTransfers:  AC + EC - CB - CD = 0
DavidTransfers:  CD + ED - DA = 0
EdwardTransfers:  AE - EC - ED = 0
BOUNDS
BA < 5
AC < 3
AE < 10
CB < 6
CD < 4
DA < 4
```

```
EC < 7
ED < 3
END
```

Solving this linear program. we find that the objective value equals 5. This means that Bill is *not* bankrupt since he can cover his total obligations ($ 5 millions owed to Albert). Using the same formulation for Charles, replacing the objective by `Min CB + CD` , we find that the maximum objective value is 9. Since Charles owes 10 altogether, he is bankrupt. Solving similar linear programs for the other three players, we find that Albert, Charles and Edward are bankrupt while Bill and David are not.

1.4.4 Arbitrage in the Currency Market

Consider the world's currency market. Given two currencies, say the Yen and the USDollar, there is an exchange rate between them (currently about 133 Yens to the Dollar). It is axiomatic of a arbitrage-free market that there is no method of converting, say, a Dollar to Yens then to Euros, then Pounds, and to Dollars so that you end up with more than a dollar. How would you recognize when there is an arbitrage possibility?

These are actual trades made on February 14, 2002.

	Dollar	Euro	Pound	Yen
Dollar		.8706	1.4279	.00750
Euro	1.1486		1.6401	.00861
Pound	.7003	.6097		.00525
Yen	133.38	116.12	190.45	

It is not obvious, but the Dollar-Pound-Yen-Dollar conversion actually makes $0.0003 per dollar converted. How would you formulate a linear program to recognize this?

Answer:

```
VARIABLES
DE = quantity of dollars changed into euros
DP = quantity of dollars changed into pounds
DY = quantity of dollars changed into yens
ED = quantity of euros changed into dollars
EP = quantity of euros changed into pounds
EY = quantity of euros changed into yens
PD = quantity of pounds changed into dollars
PE = quantity of pounds changed into euros
PY = quantity of pounds changed into yens
YD = quantity of yens changed into dollars
YE = quantity of yens changed into euros
YP = quantity of yens changed into pounds
D = quantity of dollars generated through arbitrage
OBJECTIVE
Max D
```

```
CONSTRAINTS
Dollar:  D + DE + DP + DY - 0.8706*ED - 1.4279*PD - 0.00750*YD = 1
Euro:  ED + EP + EY - 1.1486*DE - 1.6401*PE - .00861*YE = 0
Pound:  PD + PE + PY - 0.7003*DP - 0.6097*EP - 0.00525*YP = 0
Yen:  YD + YE + YP - 133.38*DY - 116.12*EY - 190.45*PY = 0
BOUNDS
D < 10000
END
```

Solving this linear program, we find that, in order to gain $10,000 in arbitrage, we have to change about $15.6 million dollars into pounds, then convert these pounds into yens and finally change the yens into dollars. There are other solutions as well.

1.4.5 Tax Clientele Effects in Bond Portfolio Management

Here, we consider a model proposed by E.I. Ronn, "A New Linear Programming approach to Bond Portfolio Management," *Journal of Financial and Quantitative Analysis 22* (1987) 439-466. See also S.M. Schaefer, "Tax Induced Clientele Effects in the Market for British Government Securities," *Journal of Financial Economics 10* (1982), 121-159.

The goal is to construct an optimal tax-specific bond portfolio, for a given tax bracket, by exploiting the price differential of an after-tax stream of cash flows. This objective is accomplished by purchasing at the ask price "underpriced" bonds (for the specific tax bracket), while simultaneously selling at the bid price "overpriced" bonds. In addition, the model can also be used to measure the after-tax term structure of spot U.S. Government interest rates for both tax-exempt and taxable investors.

The following notation is assumed:

P_j^a = asked price of bond j

P_j^b = bid price of bond j

X_j^a = amount bought of bond j

X_j^b = amount of bond j sold short, and

$J = \{1, \ldots, j, \ldots, N\}$ = set of riskless bonds.

The objective function of the program is

$$Z = max \sum_{j=1}^{N} P_j^b X_j^b - \sum_{j=1}^{N} P_j^a X_j^a \tag{1.2}$$

since the long side of an arbitrage position must be established at ask prices while the short side of the position must be established at bid prices. Now consider the future cash-flows of the portfolio.

$$C_1 = \sum_{j=1}^{N} a_j^1 X_j^a - \sum_{j=1}^{N} a_j^1 X_j^b \tag{1.3}$$

$$\text{For } t = 2, \ldots, T, \qquad C_t = (1+\rho)C_{t-1} + \sum_{j=1}^{N} a_j^t X_j^a - \sum_{j=1}^{N} a_j^t X_j^b, \tag{1.4}$$

where ρ = Exogenous riskless reinvestment rate
a_j^t = coupon and/or principal payment on bond j at time t.

For the portfolio to be riskless, we require

$$C_t \geq 0 \qquad t = 1, \ldots, T. \tag{1.5}$$

Since the bid-ask spread has been explicitly modeled, it is clear that $X_j^a \geq 0$ and $X_j^b \geq 0$ are required. Now the resulting linear program admits two possible solutions. Either all bonds are priced to within the bid-ask spread, i.e. $Z = 0$, or infinite arbitrage profits may be attained, i.e. $Z = \infty$. Clearly any attempt to exploit price differentials by taking extremely large positions in these bonds would cause price movements: the bonds being bought would appreciate in price; the bonds being sold short would decline in value. Thus, in order to provide a finite solution, the constraints $X_j^a \leq 1$ and $X_j^b \leq 1$ are imposed. Thus, with

$$0 \leq X_j^a,\ X_j^b \leq 1 \qquad j = 1, \ldots, N, \tag{1.6}$$

the complete problem is now specified as (1.2)-(1.6).

Taxes

The proposed model explicitly accounts for the taxation of income and capital gains for specific investor classes. This means that the cash flows need to be adjusted for the presence of taxes.

For a discount bond (i.e. when $P_j^a < 100$), the after-tax cash-flow of bond j in period t is given by

$$a_j^t = c_j^t(1-\tau),$$

where c_j^t is the semiannual coupon payment
and τ is the ordinary income tax rate.

At maturity, the j^{th} bond yields

$$a_j^t = (100 - P_j^a)(1-g) + P_j^a,$$

where g is the capital gains tax rate.

For premium bond (i.e. when $P_j^a > 100$), the premium is amortized against ordinary income over the life of the bond, giving rise to an after-tax coupon payment of

$$a_j^t = \left[c_j^t - \frac{P_j^a - 100}{n_j}\right](1-\tau) + \frac{P_j^a - 100}{n_j}$$

where n_j is the number of coupon payments remaining to maturity.

A premium bond also makes a nontaxable repayment of

$$a_j^t = 100$$

at maturity.

Data

The model requires that the data contain bonds with perfectly forcastable cash flows. All callable bonds are excluded from the sample. For the same reason, flower bonds of all types are excluded. Thus, all noncallable bonds and notes are deemed appropriate for inclusion in the sample.

Major categories of taxable investors are Domestic Banks, Insurance Companies, Individuals, Nonfinancial Corporations, Foreigners. In each case, one needs to distinguish the tax rates on capital gains versus ordinary income.

The fundamental question to arise from this study is: does the data reflect tax clientele effects or arbitrage opportunities?

Consider first the class of tax-exempt investors. Using current data, form the optimal "purchased" and "sold" bond portfolios. Do you observe the same tax clientele effect as documented by Schaefer for British government securities; namely, the "purchased" portfolio contains high coupon bonds and the "sold" portfolio is dominated by low coupon bonds. In other words, the preferential taxation of capital gains for (most) taxable investors causes them to gravitate towards low coupon bonds. Consequently, for tax-exempt investors, low coupon bonds are "overpriced" and not desirable as investment vehicles.

Repeat the same analysis with the different types of taxable investors. Do you observe:

1. a clientele effect in the pricing of US Government investments, with tax- exempt investors, or those without preferential treatment of capital gains, gravitating towards high coupon bonds?

2. that not all high coupon bonds are desirable to investors without preferential treatment of capital gains? Nor are all low coupon bonds attractive to those with preferential treatment of capital gains. Can you find reasons why this may be the case?

The dual price, say u_t, associated with constraint (1.4) represents the present value of an additional dollar at time t. Explain why. It follows that u_t may be used to compute the term structure of spot interest rates R_t, given by the relation

$$R_t = \left(\frac{1}{u_t}\right)^{\frac{1}{t}} - 1.$$

Compute this week's term structure of spot interest rates for tax-exempt investors.

Chapter 2

Nonlinear Programming and Portfolio Optimization

2.1 Introduction

The term *nonlinear programming* usually refers to problems such as

$$\begin{array}{ll} \textit{maximize} & f(x_1, \ldots, x_n) \\ \textit{subject to} & \\ & g_i(x_1, \ldots, x_n) \leq 0 \quad \text{for } i = 1, \ldots, m \end{array}$$

where f and g_i are functions of the n real variables $x_1, \ldots, x_n$. Linear programming is the special case when these functions are linear. There are many problems, however, where the functions involved are not all linear. In fact, it is probably true that the real–world problems that fit strictly the mold of linearity are the exception rather than the rule. Here are some illustrations:

1. **Probabilistic elements:** Nonlinearities frequently arise when some of the coefficients in the model are random variables. For example, consider a linear program where the right–hand sides are random. To illustrate, suppose the LP has two constraints:

   $$\begin{array}{ll} \textit{maximize} & c_1x_1 + \ldots + c_nx_n \\ & a_{11}x_1 + \ldots + a_{1n}x_n \leq b_1 \\ & a_{21}x_1 + \ldots + a_{2n}x_n \leq b_2 \end{array}$$

   where the coefficients b_1 and b_2 are independently distributed and $G_i(y)$ represents the probability that the random variable b_i is at least as large as y. Suppose you want to select the variable $x_1, \ldots, x_n$ so that the joint probability of both the constraints being satisfied is at least β:

   $$P[a_{11}x_1 + \ldots + a_{1n}x_n \leq b_1] \times P[a_{21}x_1 + \ldots + a_{2n}x_n \leq b_2] \geq \beta.$$

   Then this condition can be written as the following set of constraints:

   $$\begin{array}{rrcl} -y_1 & a_{11}x_1 + \ldots + a_{1n}x_n & = & 0 \\ -y_2 & a_{21}x_1 + \ldots + a_{2n}x_n & = & 0 \\ & G_1(y_1) \times G_2(y_2) & \geq & \beta, \end{array}$$

where this product leads to nonlinear restrictions on y_1 and y_2.

2. **Portfolio selection:** Financial analysts in banks and insurance companies have devoted considerable attention to mathematical models that assist in managing portfolios of common stocks, bonds and other securities. Inherent in such models is an assessment of a proposed porfolio's expected gain and the associated risk. Variance and covariance terms in the risk function typically lead to quadratic terms. We discuss portfolio optimization in Sections 2.10 and 2.11 of this chapter.

3. **Constructing an index fund:** In integer programming applications, such as a model discussed in the next chapter for constructing an index fund, the "relaxation" can be written as a multivariate function that is convex but nondifferentiable. Subgradient techniques are used to solve this class of nonlinear optimization problems.

To solve nonlinear programming problems by hand, the usual approach is to write the optimality conditions and then to solve the resulting system. Even in the case of unconstrained optimization, it must be obvious to you that this approach is not practical for more than three or four variables. The reasons are:

1. Setting the first partial derivatives equal to zero gives a system of n equations in n unknowns. Unless this system is linear (i.e. the original function was quadratic) it is not easy to find solutions. It may well be impossible to do by hand.

2. The second order sufficiency conditions are quite complicated, requiring the evaluation of determinants in the Hessian matrix. Even in the case of one or two decision variables, if the function f is sufficiently complicated, it may not be possible to hand-solve the optimality conditions, and hence this approach is not generally viable.

The situation only gets worse when the problem contains equality and/or inequality constraints. For these reasons, it seems reasonable to expect that numerical methods will be needed for most nonlinear programming problems. In contrast to linear programming, where the simplex method can handle most instances and reliable implementations are widely available, there is not a single preferred algorithm for solving nonlinear programs. Without difficulty, one can find ten or fifteen methods in the literature and the underlying theory of nonlinear programming is still evolving. A systematic comparison between methods is complicated by the fact that a nonlinear method can be very effective for one type of problem and yet fail miserably for another. In these notes, we sample a few ideas:

1. the method of steepest ascent for unconstrained optimization,

2. the generalized reduced-gradient algorithm,

3. the Lagrangian approach,

4. sequential quadratic programming,

5. subgradient optimization for nondifferentiable functions.

2.2 Software

Some software packages for solving nonlinear programs are:

1. CONOPT, GRG2, Excel's SOLVER (all three are based on the generalized reduced-gradient algorithm),
2. MINOS, LANCELOT (both based on the Lagrangian approach),
3. QL, LSSOL, QPOPT (for solving quadratic programs; In this case, the optimality conditions are linear. So, linear programming approaches are relevant: simplex method, interior point method),
4. MATLAB optimization toolbox, SNOPT, NLPQL (sequential quadratic programming).

A good source for learning about existing software is the web site
http://www-neos.mcs.anl.gov/neos/
at Argone National Labs.

Of course, as is the case for linear programming, you will need a modeling language to work efficiently with large nonlinear models. Two of the most popular are GAMS and AMPL. Most of the optimizers described above accept models written is either of these mathematical programming languages.

2.3 Line search

Before discussing optimization methods for multivariate constrained problems, we start with line search (for functions of one variable), an important component to many nonlinear programming algorithms.

2.3.1 Binary search

Binary search is a very simple idea for solving numerically $f(x) = 0$, where f is a function of a single variable.

For example, suppose we want to find the maximum of $g(x) = 2x^3 - e^x$. This is equivalent to solving the equation $g'(x) = 6x^2 - e^x = 0$. But there is no closed form solution. So we solve the equation numerically, through binary search. If we let $f(x) = 6x^2 - e^x$, we first look for two points, say a, b, such that the signs of $f(a)$ and $f(b)$ are opposite. Here $a = 0$ and $b = 1$ would do since $f(0) = -1$ and $f(1) \approx 3.3$. Since f is continuous, we know that there exists an x with $0 < x < 1$ such that $f(x) = 0$. We say that our confidence interval is [0,1]. Now let us try the middle point $x = 0.5$. $f(0.5) \approx -0.1487 < 0$ so we get the new confidence interval [0.5, 1.0]. We continue with $x = 0.75$ and since $f(0.75) > 0$ we get the confidence interval [0.5,0.75]. Repeating this, we converge very quickly to a value of x where $f(x) = 0$. Here, after 10 iterations, we are within 0.001 of the real value.

In general, if we have a confidence interval of $[a, b]$, we evaluate $f(\frac{a+b}{2})$ to cut the confidence interval in half.

Binary search is fast. It reduces the confidence interval by a factor of 2 for every iteration, so after k iterations the original interval is reduced to $(b-a) \times 2^{-k}$. A drawback is that binary

search only finds one solution. So, if g had local extrema in the above example, binary search could converge to any of them. In fact, most algorithms for nonlinear programming are subject to failure for this reason. It turns out that, in the above example, g is concave and therefore binary search finds the unique maximum.

Example 2.3.1 *Binary search can be used to compute the internal rate of return r of an investment. Mathematically, r is the interest rate that satisfies the equation*

$$\frac{F_1}{1+r} + \frac{F_2}{(1+r)^2} + \frac{F_3}{(1+r)^3} + \ldots + \frac{F_N}{(1+r)^N} - C = 0$$

where

$$\begin{aligned} F_t &= \textit{cash flow in year } t \\ N &= \textit{number of years} \\ C &= \textit{cost of the investment} \end{aligned}$$

As an example, consider a 4-year noncallable bond with a 10% coupon rate paid annually and a par value of \$1000. Such a bond has the following cash flows:

t Years from now	F_t
1	*\$ 100*
2	*100*
3	*100*
4	*1100*

Suppose this bond is now selling for \$900. Compute the yield of this bond.

The yield r of the bond is given by the equation

$$\frac{100}{1+r} + \frac{100}{(1+r)^2} + \frac{100}{(1+r)^3} + \frac{1100}{(1+r)^4} - 900 = 0$$

Let us denote by $f(r)$ the left-hand-side of this equation. We find r such that $f(r) = 0$ using binary search.

We start by finding values (a, b) such that $f(a) > 0$ and $f(b) < 0$. In this case, we expect r to be between 0 and 1. Since $f(0) = 500$ and $f(1) = -743.75$, we have our starting values.

Next, we let $c = 0.5$ (the midpoint) and calculate $f(c)$. Since $f(0.5) = -541.975$, we replace our range with $a = 0$ and $b = 0.5$ and repeat. When we continue, we get the following table of values:

Iter.	a	c	b	$f(a)$	$f(c)$	$f(b)$
1	0	0.5	1	500	-541.975	-743.75
2	0	0.25	0.5	500	-254.24	-541.975
3	0	0.125	0.25	500	24.85902	-254.24
4	0.125	0.1875	0.25	24.85902	-131.989	-254.24
5	0.125	0.15625	0.1875	24.85902	-58.5833	-131.989
6	0.125	0.140625	0.15625	24.85902	-18.2181	-58.5833
7	0.125	0.132813	0.140625	24.85902	2.967767	-18.2181
8	0.132813	0.136719	0.140625	2.967767	-7.71156	-18.2181
9	0.132813	0.134766	0.136719	2.967767	-2.39372	-7.71156
10	0.132813	0.133789	0.134766	2.967767	0.281543	-2.39372
11	0.133789	0.134277	0.134766	0.281543	-1.05745	-2.39372
12	0.133789	0.134033	0.134277	0.281543	-0.3883	-1.05745

So the yield of the bond is $r = 13.4\%$.

Of course, this routine sort of calculation is perfectly set up for calculation by computer. For example, Excel's SOLVER can search for any particular value of a function.

Golden Section Search

Golden section search is similar in spirit to binary search. It is used to compute the maximum of a function $f(x)$ defined on an interval $[a, b]$.

It assumes that

(i) f is continuous

(ii) f has a unique local maximum in the interval $[a, b]$.

The golden search method consists in computing $f(c)$ and f(d) for $a < d < c < b$.

- If $f(c) > f(d)$, the procedure is repeated with the interval (a, b) replaced by (d, b).
- If $f(c) < f(d)$, the procedure is repeated with the interval (a, b) replaced by (a, c).

Note: The name "golden section" comes from a certain choice of c and d that yields fast convergence, namely $c = a + r(b - a)$ and $d = b + r(a - b)$, where $r = \frac{\sqrt{5}-1}{2} = .618034\ldots$. This is the golden ratio, already known to the ancient greeks.

Example 2.3.2 *Find the maximum of the function* $x^5 - 10x^2 + 2x$ *in the interval* $[0, 1]$.

In this case, we begin with $a = 0$ and $b = 1$. Using golden section search, that gives $d = 0.382$ and $c = 0.618$. The function values are $f(a) = 0$, $f(d) = -0.687$, $f(c) = -2.493$, and $f(b) = -7$. Since $f(c) < f(d)$, our new range is $a = 0$, $b = .618$. Recalculating from the new range gives $d = .236$, $c = .382$ (note that our current c was our previous d: it is this reuse of calculated values that gives golden section search its speed). We repeat this process to get the following table:

Iter.	a	d	c	b	$f(a)$	$f(d)$	$f(c)$	$f(b)$
1	0	0.382	0.618	1	0	-0.6869	-2.4934	-7
2	0	0.2361	0.382	0.618	0	-0.0844	-0.6869	-2.4934
3	0	0.1459	0.2361	0.382	0	0.079	-0.0844	-0.6869
4	0	0.0902	0.1459	0.2361	0	0.099	0.079	-0.0844
5	0	0.0557	0.0902	0.1459	0	0.0804	0.099	0.079
6	0.0557	0.0902	0.1115	0.1459	0.0804	0.099	0.0987	0.079
7	0.0557	0.077	0.0902	0.1115	0.0804	0.0947	0.099	0.0987
8	0.077	0.0902	0.0983	0.1115	0.0947	0.099	0.1	0.0987
9	0.0902	0.0983	0.1033	0.1115	0.099	0.1	0.0999	0.0987
10	0.0902	0.0952	0.0983	0.1033	0.099	0.0998	0.1	0.0999
11	0.0952	0.0983	0.1002	0.1033	0.0998	0.1	0.1	0.0999
12	0.0983	0.1002	0.1014	0.1033	0.1	0.1	0.1	0.0999
13	0.0983	0.0995	0.1002	0.1014	0.1	0.1	0.1	0.1
14	0.0995	0.1002	0.1007	0.1014	0.1	0.1	0.1	0.1
15	0.0995	0.0999	0.1002	0.1007	0.1	0.1	0.1	0.1
16	0.0995	0.0998	0.0999	0.1002	0.1	0.1	0.1	0.1
17	0.0998	0.0999	0.1	0.1002	0.1	0.1	0.1	0.1
18	0.0999	0.1	0.1001	0.1002	0.1	0.1	0.1	0.1
19	0.0999	0.1	0.1	0.1001	0.1	0.1	0.1	0.1
20	0.0999	0.1	0.1	0.1	0.1	0.1	0.1	0.1
21	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

Again we can use SOLVER to maximize this function directly.

2.4 Unconstrained Optimization: Optimality Conditions

Many of the concepts for functions of one variable can be extended to functions of several variables. For example, the gradient extends the notion of derivative, the Hessian matrix that of second derivative, etc.

2.4.1 Gradient

Given a function f of n variables $x_1, x_2, \ldots, x_n$, we define the *partial derivative* relative to variable x_i, written as $\frac{\partial f}{\partial x_i}$, to be the derivative of f with respect to x_i treating all variables except x_i as constant. Let x denote the vector $(x_1, x_2, \ldots, x_n)$. With this notation, $f(x) = f(x_1, x_2, \ldots, x_n)$, $\frac{\partial f}{\partial x_i}(x) = \frac{\partial f}{\partial x_i}(x_1, x_2, \ldots, x_n)$, etc. The *gradient* of f at x, written $\nabla f(x)$, is the vector $\begin{pmatrix} \frac{\partial f}{\partial x_1}(x) \\ \frac{\partial f}{\partial x_2}(x) \\ \vdots \\ \frac{\partial f}{\partial x_n}(x) \end{pmatrix}$. The gradient vector $\nabla f(x)$ gives the direction of steepest ascent of the function f at point x. The gradient acts like the derivative in that small changes around a given point x^* can be estimated using the gradient.

$$f(x^* + \Delta) \approx f(x^*) + \Delta \nabla f(x^*)$$

where $\Delta = (\Delta_1, \ldots, \Delta_n)$ denotes the vector of changes.

Example 2.4.1 *If* $f(x_1, x_2) = x_1^2 - 3x_1x_2 + x_2^2$, *then* $f(1,1) = -1$. *What about* $f(1.01, 1.01)$*?*

In this case, $x^* = (1,1)$ and $\Delta = (0.01, 0.01)$. Since $\frac{\partial f}{\partial x_1}(x_1, x_2) = 2x_1 - 3x_2$ and $\frac{\partial f}{\partial x_2}(x_1, x_2) = -3x_1 + 2x_2$, we get

$$\nabla f(1,1) = \begin{pmatrix} -1 \\ -1 \end{pmatrix}.$$

So $f(1.01, 1.01) = f((1,1)+(0.01, 0.01)) \approx f(1,1)+(0.01, 0.01)\nabla f(1,1) = -1+(0.01, 0.01)\begin{pmatrix} -1 \\ -1 \end{pmatrix}$ $= -1.02$.

2.4.2 Hessian matrix

Second partials $\frac{\partial^2 f}{\partial x_i \partial x_j}(x)$ are obtained from $f(x)$ by taking the derivative relative to x_i (this yields the first partial $\frac{\partial f}{\partial x_i}(x)$) and then by taking the derivative of $\frac{\partial f}{\partial x_i}(x)$ relative to x_j. So we can compute $\frac{\partial^2 f}{\partial x_1 \partial x_1}(x)$, $\frac{\partial^2 f}{\partial x_1 \partial x_2}(x)$ and so on. These values are arranged into the *Hessian* matrix

$$H(x) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1}(x) & \frac{\partial^2 f}{\partial x_1 \partial x_2}(x) & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n}(x) \\ \frac{\partial^2 f}{\partial x_2 \partial x_1}(x) & \frac{\partial^2 f}{\partial x_2 \partial x_2}(x) & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n}(x) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1}(x) & \frac{\partial^2 f}{\partial x_n \partial x_2}(x) & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n}(x) \end{pmatrix}$$

The Hessian matrix is a symmetric matrix, that is $\frac{\partial^2 f}{\partial x_i \partial x_j}(x) = \frac{\partial^2 f}{\partial x_j \partial x_i}(x)$.

2.4.3 Maximum and Minimum

Optima can occur in three places:

1. at the boundary of the domain,
2. at a nondifferentiable point, or
3. at a point x^* with $\nabla f(x^*) = 0$.

We will identify the first type of point with Kuhn–Tucker conditions (see Section 2.7.2). The second type is found only by ad hoc methods. The third type of point can be found by solving the gradient equations $\nabla f(x^*) = 0$. To identify if a point x^* with zero gradient is a local maximum or local minimum, check the Hessian:

- If $H(x^*)$ is positive definite then x^* is a local minimum.

- If $H(x^*)$ is negative definite, then x^* is a local maximum.

A square matrix A is *positive definite* if $x^T Ax > 0$ for all nonzero column vectors x. It is *negative definite* if $x^T Ax < 0$ for all nonzero x. These definitions are hard to check directly. More useful in practice are the following properties, which hold when the matrix A is symmetric (that is the case of interest to us), and which are easier to check.

The ith *leading principal minor* of A is the matrix A_i formed by the first i rows and columns of A. So, the first leading principal minor of A is the matrix $A_1 = (a_{11})$, the second leading principal minor is the matrix $A_2 = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$, and so on.

- The matrix A is positive definite if all its leading principal minors A_1, $A_2, \ldots A_n$ have strictly positive determinants.

- If these determinants are nonzero and alternate in signs, starting with $\det(A_1) < 0$, then the matrix A is negative definite.

Example 2.4.2 *Find the local extrema of* $f(x_1, x_2) = x_1^3 + x_2^3 - 3x_1x_2$.

This function is everywhere differentiable, so extrema can only occur at points x^* such that $\nabla f(x^*) = 0$.

$$\nabla f(x) = \begin{pmatrix} 3x_1^2 - 3x_2 \\ 3x_2^2 - 3x_1 \end{pmatrix}$$

This equals 0 iff $(x_1, x_2) = (0, 0)$ or $(1, 1)$. The Hessian is

$$H(x) = \begin{pmatrix} 6x_1 & -3 \\ -3 & 6x_2 \end{pmatrix}$$

So,

$$H(0,0) = \begin{pmatrix} 0 & -3 \\ -3 & 0 \end{pmatrix}$$

Let II_1 denote the first principal minor of $II(0,0)$ and let II_2 denote its second principal minor. Then $\det(H_1) = 0$ and $\det(H_2) = -9$. Therefore $H(0,0)$ is neither positive nor negative definite.

$$H(1,1) = \begin{pmatrix} 6 & -3 \\ -3 & 6 \end{pmatrix}$$

Its first principal minor has $\det(H_1) = 6 > 0$ and its second principal minor has $\det(H_2) = 36 - 9 = 25 > 0$. Therefore $H(1,1)$ is positive definite, which implies that $(1,1)$ is a local minimum.

2.4.4 Global Optima

Finding global maxima and minima is harder. There is one case that is of interest.

We say that a domain is *convex* if every line drawn between two points in the domain lies within the domain.

We say that a function f is *convex* if the line connecting any two points lies above the function. That is, for all x, y in the domain and $0 < \alpha < 1$, we have

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y).$$

- If a function f is convex on a convex domain, and $\nabla f(x) = 0$ has a solution x^* in the domain, then x^* is a global minimum.
- If a function f is concave on a convex domain, and $\nabla f(x) = 0$ has a solution x^* in the domain, then x^* is a global maximum.

Is there an easy way to check that a function f is convex or concave? The answer is to compute the Hessian matrix $H(x)$. If $H(x)$ is positive definite for every point x in the domain, then f is convex. If $H(x)$ is negative definite for every x in the domain, then f is concave.

Actually, the concepts of positive definite and negative definite matrices need to be extended for our purpose. Let A be an $n \times n$ symmetric matrix. An i^{th} *principal minor* of A is an $i \times i$ matrix obtained from A by deleting $n - i$ rows and the corresponding $n - i$ columns.

- The matrix A is *positive semidefinite* if all its principal minors have nonnegative determinants.
- The matrix A is *negative semidefinite* if the determinants of all i^th principal minors are less than or equal to 0 when i is odd, and greater than or equal to 0 when i is even.

Positive semidefinite and negative semidefinite matrices are relevant here for the following reason.

f is convex on domain D if and only if its Hessian $H(x)$ is positive semidefinite for all x in D.

f is concave on domain D if and only if its Hessian $H(x)$ is negative semidefinite for all x in D.

Example 2.4.3 *Show that the function* $f(x_1, x_2, x_3) = x_1^4 + (x_1 + x_2)^2 + (x_1 + x_3)^2$ *is convex over* $\Re^3$.

$\frac{\partial f}{\partial x_1}(x) = 4x_1^3 + 2(x_1 + x_2) + 2(x_1 + x_3)$
$\frac{\partial f}{\partial x_2}(x) = 2(x_1 + x_2)$
$\frac{\partial f}{\partial x_3}(x) = 2(x_1 + x_3)$

$$H(x_1, x_2, x_3) = \begin{pmatrix} 12x_1^2 + 4 & 2 & 2 \\ 2 & 2 & 0 \\ 2 & 0 & 2 \end{pmatrix}$$

The determinants of the three 1×1 principal minors are $12x_1^2 + 4$, 2 and 2 (each is positive), those of the three 2×2 principal minors are $12x_1^2 \geq 0$, $12x_1^2 \geq 0$ and 4, and the determinant of the 3×3 principal minor is $48x_1^2 \geq 0$. So $H(x_1, x_2, x_3)$ is positive semidefinite for all (x_1, x_2, x_3) in $\Re^3$. This implies that f is convex over $\Re^3$. This conclusion also follows from noting that f is the sum of convex functions.

2.5 The method of steepest ascent (or descent)

In this section, we consider unconstrained nonlinear programs of the form

maximize $f(\mathbf{x})$, where $\mathbf{x} = (x_1, \ldots, x_n)$.

The simplest numerical method for finding a solution is based on the idea of going uphill on the graph of the function f. The gradient of a function always points in the direction of fastest increase. This suggests that, if our current estimate of the maximizing point is $\mathbf{x}^c$, we should move in the direction of $\nabla f(\mathbf{x}^c)$. Now that we have a direction, deciding how far we should go in this direction is just a line search. It can be performed by binary search or by golden section search. This will provide a new estimate of the maximizing point and the procedure can be repeated.

We illustrate this approach on the following example:

minimize $f(\mathbf{x}) = (x_1 - 2)^4 + (x_1 - 2x_2)^2$.

The first step is to compute the gradient.

$\nabla f(\mathbf{x}) = \left(\; 4(x_1 - 2)^3 + 2(x_1 - 2x_2), -4(x_1 - 2x_2) \; \right)$.

Next we need to choose a starting point. We arbitrarily select the point $\mathbf{x}^0 = (0, 3)$. Now we are ready to compute the steepest descent direction at point $\mathbf{x}^0$. It is the direction opposite to the gradient vector computed at $\mathbf{x}^0$, namely

$\mathbf{d^0} = -\nabla f(\mathbf{x}^0) = (44, -24)$.

If we move from $\mathbf{x}^0$ in the direction $\mathbf{d}^0$, we get a new point $\mathbf{x}^0 + \alpha\mathbf{d}^0$. where α determines how far we go ($\alpha = 0$ corresponds to staying at $\mathbf{x}^0$). Since our goal is to minimize f, we should move to a point $\mathbf{x}^1 = \mathbf{x}^0 + \alpha\mathbf{d}^0$ where α is chosen accordingly. The optimal value of α can be found by solving the following one–dimensional minimization problem:

$$\begin{aligned} \min f(\mathbf{x}^0 + \alpha\mathbf{d}^0) &= ([0 + 44\alpha] - 2)^4 + ([0 + 44\alpha] - 2[3 - 24\alpha])^2 \\ &= 3748096\alpha^4 - 681472\alpha^3 + 54928\alpha^2 - 2512\alpha + 52. \end{aligned}$$

This minimization can be performed through a numerical line search procedure such as the golden section search. The main difficulty with using most line search procedures is that it is necessary to specify an interval $[a, b]$ over which to conduct the search. In our example a could obviously be chosen equal to zero, but what should we select as b? One way is to start with a trial value of $0.01\|\mathbf{d}^0\|$ where $\|\mathbf{d}^0\|$ is the length of vector $\mathbf{d}^0$. Then we try $0.02\|\mathbf{d}^0\|$, $0.04\|\mathbf{d}^0\|$, $0.08\|\mathbf{d}^0\|$ and so on. As soon as the function value goes down and then up, we stop incrementing and select b as the last trial and a as the previous one.

In our example, the line search yields $\alpha = 0.062$ as the optimum value. Therefore we get $\mathbf{x}^1 = \mathbf{x}^0 + \alpha\mathbf{d}^0 = (0, 3) + 0.062(44, -24) = (2.70, 1.51)$.

Now we repeat the process starting from this point, by calculating the search direction $\mathbf{d^1} = -\nabla f(\mathbf{x}^1)$, conducting a line search in that direction and so forth. The iterations

continue until some stopping creterion is met. For example, we might stop when $\|\nabla f(\mathbf{x}^k)\|$ is close enough to zero, or after we have performed a given number of iterations. Another possible stopping criterion is that the successive values of $\mathbf{x}^k$ are close enough together, say $\|\mathbf{x}^{k+1} - \mathbf{x}^k\| < T$ where T is a tolerance parameter. Using $T = 0.05$, the algorithm stops after determining $\mathbf{x}^5$. The results are summarized in the next table.

k	(x_1^k, x_2^k)	(d_1^k, d_2^k)	α^k	$\|\mathbf{x}^{k+1} - \mathbf{x}^k\|$
0	(0.00, 3.00)	(44.00, -24.0)	0.062	3.08
1	(2.70, 1.51)	(-0.73, -1.28)	0.24	0.36
2	(2.52, 1.20)	(-0.80, 0.48)	0.11	0.10
3	(2.43, 1.25)	(-0.18, -0.28)	0.31	0.11
4	(2.37, 1.16)	(-0.30, 0.20)	0.12	0.04
5	(2.33, 1.18)			

If you plot the points $\mathbf{x}^0$, $\mathbf{x}^1, \ldots, \mathbf{x}^5$ in two dimensions, and draw lines between $\mathbf{x}^i$ and $\mathbf{x}^{i+1}$, you will observe the *zigzagging* phenomenon. When we pursue the steepest descent algorithm for more iterations, the zigzagging phenomenon becomes even more pronounced and the method is slow to converge to the optimal solution $\mathbf{x}^* = (2, 1)$. There are several numerical techniques for modifying the method of steepest ascent that reduce the approach's propensity to zigzag, and thereby speed up convergence. One such technique, named the *Newton-Raphson method* after the scientists who proposed it, employs a direction based on a quadratic fit of the objective function. Specifically, the direction $\mathbf{d}^k$ is found by solving the linear system

$\sum_{j=1}^{n} \frac{\partial^2 f}{\partial x_i \partial x_j} = -\frac{\partial f}{\partial x_i}$ for $i = 1, \ldots, n$.

Because the Hessian matrix is expensive to compute at each iteration, an approximation may be used instead. Such approaches are known as quasi-Newton methods. Other acceleration approaches have been developed but we will not go into them here.

2.6 The generalized reduced gradient method

First we consider an example where the constraints are linear equations.

$$\begin{array}{lrll} \textit{minimize} & f(\mathbf{x}) & = & x_1^2 + x_2 + x_3^2 + x_4 \\ & g_1(\mathbf{x}) & = & x_1 + x_2 + 4x_3 + 4x_4 - 4 \quad = 0 \\ & g_2(\mathbf{x}) & = & -x_1 + x_2 + 2x_3 - 2x_4 + 2 \quad = 0. \end{array}$$

It is easy to solve the constraint equations for two of the variables in terms of the others. Solving for x_2 and x_3 in terms of x_1 and x_4 gives

$$x_2 = 3x_1 + 8x_4 - 8 \text{ and } x_3 = -x_1 - 3x_4 + 3.$$

Substituting these expressions into the objective function yields the following reduced problem:

$\textit{minimize} \quad f(x_1, x_4) = x_1^2 + (3x_1 + 8x_4 - 8) + (-x_1 - 3x_4 + 3)^2 + x_4.$

This problem is unconstrained and therefore it can be solved by the method of steepest descent (see previous section).

Now consider the possibility of approximating a problem where the constraints are nonlinear equations by a problem with linear equations, which can then be solved like the preceding example. To see how this works, consider the following example, which resembles the preceding one but has constraints that are nonlinear.

$$\begin{array}{lrcll} \textit{minimize} & f(\mathbf{x}) & = & x_1^2 + x_2 + x_3^2 + x_4 & \\ & g_1(\mathbf{x}) & = & x_1^2 + x_2 + 4x_3 + 4x_4 - 4 & = 0 \\ & g_2(\mathbf{x}) & = & -x_1 + x_2 + 2x_3 - 2x_4^2 + 2 & = 0. \end{array}$$

We use the following approximation, seen earlier:

$$g(\mathbf{x}) \approx g(\bar{\mathbf{x}}) + \nabla g(\bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^T.$$

This gives

$$\begin{aligned} g_1(\mathbf{x}) &\approx (\bar{x}_1^2 + \bar{x}_2 + 4\bar{x}_3 + 4\bar{x}_4 - 4) + (2\bar{x}_1, 1, 4, 4) \begin{pmatrix} x_1 - \bar{x}_1 \\ x_2 - \bar{x}_2 \\ x_3 - \bar{x}_3 \\ x_4 - \bar{x}_4 \end{pmatrix} \\ &\approx 2\bar{x}_1 x_1 + x_2 + 4x_3 + 4x_4 - (\bar{x}_1^2 + 4) = 0 \end{aligned}$$

and

$$g_2(\mathbf{x}) \approx -x_1 + x_2 + 2x_3 - 4\bar{x}_4 x_4 + (\bar{x}_4^2 + 2) = 0.$$

The idea of the *generalized reduced gradient algorithm (GRG)* is to solve a sequence of subproblems, each of which uses a linear approximation of the constraints. In each iteration of the algorithm, the constraint linearization is recalculated at the point found from the previous iteration. Typically, even though the constraints are only approximated, the subproblems yield points that are progressively closer to the optimal point. A property of the linearization is that, at the optimal point, the linearized problem has the same solution as the original problem.

The first step in applying GRG is to pick a starting point. Suppose that we start with $\mathbf{x}^0 = (0, -8, 3, 0)$, which happens to satisfy the original constraints. It is possible to start from an infeasible point, but the details of how to do that need not concern us until later. Using the approximation formulas derived earlier, we form our first approximation problem as follows.

$$\begin{array}{lrcll} \textit{minimize} & f(\mathbf{x}) & = & x_1^2 + x_2 + x_3^2 + x_4 & \\ & g_1(\mathbf{x}) & = & x_2 + 4x_3 + 4x_4 - 4 & = 0 \\ & g_2(\mathbf{x}) & = & -x_1 + x_2 + 2x_3 + 2 & = 0. \end{array}$$

Now we solve the equality constraints of the approximate problem to express two of the variables in terms of the others. Arbitrarily selecting x_2 and x_3, we get

$$x_2 = 2x_1 + 4x_4 - 8 \text{ and } x_3 = -\frac{1}{2}x_1 - 2x_4 + 3.$$

Substituting these expressions in the objective function yields the reduced problem

$$min\ f(x_1, x_4) = x_1^2 + (2x_1 + 4x_2 - 8) + (-\tfrac{1}{2}x_1 - 2x_4 + 3)^2 + x_4.$$

Solving this unconstrained minimization problem yields $x_1 = -0.375$, $x_4 = 0.96875$. Substituting in the equations for x_2 and x_3 gives $x_2 = -4.875$ and $x_3 = 1.25$. Thus the first iteration of GRG has produced the new point $\mathbf{x}^1 = (-0.375, -4.875, 1.25, 0.96875)$.

To continue the solution process, we would relinearize the constraint functions at the new point, use the resulting system of linear equations to express two of the variables in terms of the others, substitute into the objective to get the new reduced problem, solve the reduced problem for $\mathbf{x}^2$, and so forth. Using the stopping criterion $\|\mathbf{x}^{k+1}-\mathbf{x}^k\| < T$ where $T = 0.0025$, we get the results summarized in Table 2.1.

k	$(x_1^k, x_2^k, x_3^k, x_4^k)$	$f(\mathbf{x}^k)$	$\|\mathbf{x}^{k+1} - \mathbf{x}^k\|$
0	(0.000, -8.000, 3.000, 0.000)	1.000	3.729
1	(-0.375, -4.875, 1.250, 0.969)	-2.203	0.572
2	(-0.423, -5.134, 1.619, 0.620)	-1.714	0.353
3	(-0.458, -4.792, 1.537, 0.609)	-1.610	0.022
4	(-0.478, -4.802, 1.534, 0.610)	-1.611	0.015
5	(-0.488, -4.813, 1.534, 0.610)	-1.612	0.008
6	(-0.494, -4.818, 1.534, 0.610)	-1.612	0.004
7	(-0.497, -4.821, 1.534, 0.610)	-1.612	0.002
8	(-0.498, -4.823, 1.534, 0.610)	-1.612	

Table 2.1: Summarized results

This is to be compared with the optimum solution which is

$$x^* = (-0.500, -4.825, 1.534, 0.610).$$

Note that, in Table 2.1, the values of the function $f(x^k)$ are sometimes smaller than the minimum value, which is -1.612! How is this possible? The reason is that the points x^k computed by GRG are usually not feasible to the constraints. They are only feasible to a linear approximation of these constraints.

Now we discuss the method used by GRG for starting at an infeasible solution: a *phase 1 problem* is solved to construct a feasible one. The objective function for the phase 1 problem is the sum of the absolute values of the violated constraints. The constraints for the phase 1 problem are the nonviolated ones. Suppose we had started at the point $\mathbf{x}^0 = (1, 1, 0, 1)$ in our example. This point violates the first constraint but satisfies the second, so the phase 1 problem would be

$$\begin{array}{ll} \textit{minimize} & |x_1^2 + x_2 + 4x_3 + 4x_4 - 4| \\ & -x_1 + x_2 + 2x_3 - 2x_4^2 + 2 \quad = 0. \end{array}$$

Once a feasible solution has been found by solving the phase 1 problem, the method illustrated above is used to find an optimal solution.

Finally, we discuss how GRG solves problems having inequality constraints as well as equalities. At each iteration, only the tight inequality constraints enter into the system of linear equations used for eliminating variables (these inequality constraints are said to be *active*). The process is complicated by the fact that active inequality constraints at the current point may need to be released in order to move to a better solution. We illustrate the ideas on the following example.

$$\begin{array}{lll} \textit{minimize} & f(x_1, x_2) = & (x_1 - \frac{1}{2})^2 + (x_2 - \frac{5}{2})^2 \\ & & x_1 - x_2 \geq 0 \\ & & x_1 \geq 0 \\ & & 0 \leq x_2 \leq 2. \end{array}$$

The first step in applying GRG is to pick a starting point. Suppose that we start from $\mathbf{x}^0 = (1, 0)$. This point satisfies all the constraints: $x_1 - x_2 \geq 0$, $x_1 \geq 0$ and $x_2 \leq 2$ are inactive, whereas the constraint $x_2 \geq 0$ is active. We have to decide whether x_2 should stay at its lower bound or be allowed to leave its bound.

$$\nabla f(\mathbf{x}^0) = (2x_1^0 - 1, 2x_2^0 - 5) = (1, -5).$$

This indicates that we will get the largest decrease in f if we move in the direction $\mathbf{d}^0 = -\nabla f(\mathbf{x}^0) = (-1, 5)$, i.e. if we decrease x_1 and increase x_2. Since this direction is towards the interior of the feasible region, we decide to release x_2 from its bound. The new point will be $\mathbf{x}^1 = \mathbf{x}^0 + \alpha^0 \mathbf{d}^0$, for some $\alpha^0 > 0$. The constraints of the problem induce an upper bound on α^0, namely $\alpha^0 \leq 0.8333$. Now we perform a line search to determine the best value of α^0 in this range. It turns out to be $\alpha^0 = 0.8333$, so $\mathbf{x}^1 = (0.8333, 0.8333)$. Now, we repeat the process: the constraint $x_1 - x_2 \geq 0$ is active whereas the others are inactive. Since the active constraint is not a simple upper or lower bound constraint, we introduce a surplus variable, say x_3, and solve for one of the variables in terms of the others. Substituting $x_1 = x_2 + x_3$, we obtain the reduced optimization problem

$$\begin{array}{lll} \textit{minimize} & f(x_2, x_3) = & (x_2 + x_3 - \frac{1}{2})^2 + (x_2 - \frac{5}{2})^2 \\ & & 0 \leq x_2 \leq 2 \\ & & x_3 \geq 0. \end{array}$$

The reduced gradient is

$$\begin{aligned} \nabla f(x_2, x_3) &= (2x_2 + 2x_3 - 1 + 2x_2 - 5, 2x_2 + 2x_3 - 1) \\ &= (-2.667, 0.667) \text{ at point } (x_2, x_3)^1 = (0.8333, 0). \end{aligned}$$

Therefore, the largest decrease in f occurs in the direction (2.667, -0.667), that is when we increase x_2 and decrease x_3. But x_3 is already at its lower bound, so we cannot decrease it. Consequently, we keep x_3 at its bound, i.e. we move in the direction $\mathbf{d}^1 = (2.667, 0)$ to a new point $(x_2, x_3)^2 = (x_2, x_3)^1 + \alpha^1 \mathbf{d}^1$. A line search in this direction yields $\alpha^1 = 0.25$ and $(x_2, x_3)^2 = (1.5, 0)$. The same constraints are still active so we may stay in the space of variables x_2 and x_3. Since

$$\nabla f(x_2, x_3) = (0, 2) \text{ at point } (x_2, x_3)^2 = (1.5, 0)$$

is perpendicular to the boundary line at the current solution $\mathbf{x}^2$ and points towards the exterior of the feasible region, no further decrease in f is possible. We have found the optimal solution. In the space of original variables, this optimal solution is $x_1 = 1.5$ and $x_2 = 1.5$.

This is how some of the most widely distributed nonlinear programming solvers, such as Excel's SOLVER, GINO, CONOPT, GRG2 and several others, solves nonlinear programs, with just a few additional details such as the Newton-Raphson direction for line search (we briefly mentioned this approach in the previous section). Compared with linear programs, the problems that can be solved are significantly smaller and the solutions produced may not be very accurate. So you need to be much more cautious when interpreting the output of a nonlinear program.

2.7 Lagrangian Approach

2.7.1 Equality Constraints

Suppose we have a problem:

Maximize $5 - (x_1 - 2)^2 - 2(x_2 - 1)^2$
subject to
$x_1 + 4x_2 = 3$

If we ignore the constraint, we get the solution $x_1 = 2, x_2 = 1$, which is too large for the constraint. Let us penalize ourselves λ for making the constraint too big. We end up with a function

$$L(x_1, x_2, \lambda) = 5 - (x_1 - 2)^2 - 2(x_2 - 1)^2 + \lambda(3 - x_1 - 4x_2)$$

This function is called the *Lagrangian* of the problem. The main idea is to adjust λ so that we use exactly the right amount of the resource.

$\lambda = 0$ leads to $(2, 1)$.

$\lambda = 1$ leads to $(3/2, 0)$ which uses too little of the resource.

$\lambda = 2/3$ gives $(5/3, 1/3)$ and the constraint is satisfied exactly.

We now explore this idea more formally. Given a nonlinear program (P) with equality constraints:

Minimize (or maximize) $f(x)$
subject to
$g_1(x) = b_1$
$g_2(x) = b_2$
$\vdots$
$g_m(x) = b_m$

a solution can be found using the *Lagrangian*:

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i (b_i - g_i(x))$$

Each λ_i gives the price associated with constraint i.

The reason L is of interest is the following:

Assume $x^* = (x_1^*, x_2^*, \ldots, x_n^*)$ maximizes or minimizes $f(x)$ subject to the constraints $g_i(x) = b_i$, for $i = 1, 2, \ldots, m$. Then either

(i) the vectors $\nabla g_1(x^*), \nabla g_2(x^*), \ldots, \nabla g_m(x^*)$ are linearly dependent, or

(ii) there exists a vector $\lambda^* = (\lambda_1^*, \lambda_2^*, \ldots, \lambda_m^*)$ such that $\nabla L(x^*, \lambda^*) = 0$. I.e.

$$\frac{\partial L}{\partial x_1}(x^*, \lambda^*) = \frac{\partial L}{\partial x_2}(x^*, \lambda^*) = \cdots = \frac{\partial L}{\partial x_n}(x^*, \lambda^*) = 0$$

$$\text{and} \qquad \frac{\partial L}{\partial \lambda_1}(x^*, \lambda^*) = \cdots = \frac{\partial L}{\partial \lambda_m}(x^*, \lambda^*) = 0$$

Of course, Case (i) above cannot occur when there is only one constraint. The following example shows how it might occur.

Example 2.7.1
Minimize $x_1 + x_2 + x_3^2$
subject to
$x_1 = 1$
$x_1^2 + x_2^2 = 1$.

It is easy to check directly that the minimum is acheived at $(x_1, x_2, x_3) = (1, 0, 0)$. The associated Lagrangian is

$$L(x_1, x_2, x_3, \lambda_1, \lambda_2) = x_1 + x_2 + x_3^2 + \lambda_1(1 - x_1) + \lambda_2(1 - x_1^2 - x_2^2).$$

Observe that

$$\frac{\partial L}{\partial x_2}(1, 0, 0, \lambda_1, \lambda_2) = 1 \qquad \text{for all } \lambda_1, \lambda_2,$$

and consequently $\frac{\partial L}{\partial x_2}$ *does not* vanish at the optimal solution. The reason for this is the following. Let $g_1(x_1, x_2, x_3) = x_1$ and $g_2(x_1, x_2, x_3) = x_1^2 + x_2^2$ denote the left hand sides of the constraints. Then $\nabla g_1(1, 0, 0) = (1, 0, 0)$ and $\nabla g_2(1, 0, 0) = (2, 0, 0)$ are linearly dependent vectors. So Case (i) occurs here!

Nevertheless, Case (ii) is the fundamental case. For example, if $f(x)$ is concave and all of the $g_i(x)$ are linear, then any feasible x^* with a corresponding λ^* making $\nabla L(x^*, \lambda^*) = 0$ maximizes $f(x)$ subject to the constraints. Similarly, if $f(x)$ is convex and each $g_i(x)$ is linear, then any x^* with a λ^* making $\nabla L(x^*, \lambda^*) = 0$ minimizes $f(x)$ subject to the constraints.

Economic Interpretation

The values λ_i^* have an important economic interpretation: If the right hand side b_i of Constraint i is increased by Δ, then the optimum objective value increases by approximately $\lambda_i^* \Delta$.

In particular, consider the problem

Maximize $p(x)$
subject to
$g(x) = b$,

where $p(x)$ is a profit to maximize and b is a limited amount of resource. Then, the optimum Lagrange multiplier λ^* is the marginal value of the resource. Equivalently, if b were increased by Δ, profit would increase by $\lambda^*\Delta$. This is an important result to remember.

Similarly, if

Minimize $c(x)$
subject to
$d(x) = b$,

represents the minimum cost $c(x)$ of meeting some demand b, the optimum Lagrange multiplier λ^* is the marginal cost of meeting the demand.

Example 2.7.2 *Suppose we have a refinery that must ship finished goods to some storage tanks. Suppose further that there are two pipelines, A and B, to do the shipping. The cost of shipping x units on A is ax^2; the cost of shipping y units on B is by^2, where $a > 0$ and $b > 0$ are given. How can we ship Q units while minimizing cost? What happens to the cost if Q increases by $r\%$?*

Minimize $ax^2 + by^2$
Subject to
$x + y = Q$

$$L(x, y, \lambda) = ax^2 + by^2 + \lambda(Q - x - y)$$
$$\frac{\partial L}{\partial x}(x^*, y^*, \lambda^*) = 2ax^* - \lambda^* = 0$$
$$\frac{\partial L}{\partial y}(x^*, y^*, \lambda^*) = 2by^* - \lambda^* = 0$$
$$\frac{\partial L}{\partial \lambda}(x^*, y^*, \lambda^*) = Q - x^* - y^* = 0$$

The first two constraints give $x^* = \frac{b}{a}y^*$, which leads to

$$x^* = \frac{bQ}{a+b}, \quad y^* = \frac{aQ}{a+b}, \quad \lambda^* = \frac{2abQ}{a+b}$$

and cost of $\frac{abQ^2}{a+b}$. The Hessian matrix $H(x_1, x_2) = \begin{pmatrix} 2a & 0 \\ 0 & 2b \end{pmatrix}$ is positive definite since $a > 0$ and $b > 0$. So this solution minimizes cost, given a, b, Q.

If Q increases by $r\%$, then the RHS of the constraint increases by $\Delta = rQ$ and the minimum cost increases by $\lambda^*\Delta = \frac{2abrQ^2}{a+b}$. That is, the minimum cost increases by $2r\%$.

Example 2.7.3 How should one divide his/her savings between three mutual funds with expected returns 10%, 10% and 15% repectively, so as to minimize risk while achieving an expected return of 12%. We measure risk as the variance of the return on the investment: when a fraction x of the savings is invested in Fund 1, y in Fund 2 and z in Fund 3, where $x + y + z = 1$, the variance of the return has been calculated to be

$$400x^2 + 800y^2 + 200xy + 1600z^2 + 400yz.$$

So your problem is

$$\begin{array}{lrcrcrcrcr} \min & 400x^2 & + & 800y^2 & + & 200xy & + & 1600z^2 & + & 400yz \\ s.t. & x & + & y & + & 1.5z & = & 1.2 & & \\ & x & + & y & + & z & = & 1 & & \end{array}$$

Using the Lagrangian method, the following optimal solution was obtained

$$x = 0.5 \quad y = 0.1 \quad z = 0.4 \quad \lambda_1 = 1800 \quad \lambda_2 = -1380$$

where λ_1 is the Lagrange multiplier associated with the first constraint and λ_2 with the second constraint. The corresponding objective function value (i.e. the variance on the return) is 390. If an expected return of 12.5% was desired (instead of 12%), what would be (approximately) the correcponding variance of the return?

Since $\Delta = 0.05$, the variance would increase by

$$\Delta\lambda_1 = 0.05 \times 1800 = 90.$$

So the answer is 390+90=480.

2.7.2 Equality and Inequality Constraints

How do we handle both equality and inequality constraints in (P)? Let (P) be:

Maximize $f(x)$
Subject to
$g_1(x) = b_1$
$\vdots$
$g_m(x) = b_m$
$h_1(x) \leq d_1$
$\vdots$
$h_p(x) \leq d_p$

If you have a program with $\geq$ constraints, convert it into $\leq$ by multiplying by -1. Also convert a minimization to a maximization.

The Lagrangian is

$$L(x, \lambda, \mu) = f(x) + \sum_{i=1}^{m} \lambda_i (b_i - g_i(x)) + \sum_{j=1}^{p} \mu_j (d_j - h_j(x))$$

The fundamental result is the following:

Assume $x^* = (x_1^*, x_2^*, \ldots, x_n^*)$ maximizes $f(x)$ subject to the constraints $g_i(x) = b_i$, for $i = 1, 2, \ldots, m$ and $h_j(x) \leq d_j$, for $j = 1, 2, \ldots, p$. Then either

(i) the vectors $\nabla g_1(x^*), \ldots, \nabla g_m(x^*)$, $\nabla h_1(x^*), \ldots, \nabla h_p(x^*)$ are linearly dependent, or

(ii) there exists vectors $\lambda^* = (\lambda_1^*, \ldots, \lambda_m^*)$ and $\mu^* = (\mu_1^*, \ldots, \mu_p^*)$ such that

$$\nabla f(x^*) - \sum_{i=1}^{m} \lambda_i^* \nabla g_i(x^*) - \sum_{j=1}^{p} \mu_j^* \nabla h_j(x^*) = 0$$

$$\mu_j^*(h_j(x^*) - d_j) = 0 \quad \text{(Complementarity)}$$

$$\mu_j^* \geq 0$$

Again, Case (ii) is the fundamental case.

In general, to solve these equations, you begin with complementarity and note that either μ_j^* must be zero or $h_j(x^*) - d_j = 0$. Based on the various possibilities, you come up with one or more candidate solutions. If there is an optimal solution, then one of your candidates will be it.

The above conditions are called the *Kuhn–Tucker (or Karush–Kuhn–Tucker)* conditions. Why do they make sense?

For x^* optimal, some of the inequalities will be tight and some not. Those not tight can be ignored (and will have corresponding price $\mu_j^* = 0$). Those that are tight can be treated as equalities which leads to the previous Lagrangian stuff. So

$$\mu_j^*(h_j(x^*) - d_j) = 0 \quad \text{(Complementarity)}$$

forces either the price μ_j^* to be 0 or the constraint to be tight.

Example 2.7.4 *Minimize* $(x - 2)^2 + 2(y - 1)^2$
Subject to
$x + 4y \leq 3$
$x \geq y$

First we convert to standard form, to get

Maximize $-(x - 2)^2 - 2(y - 1)^2$
Subject to
$x + 4y \leq 3$
$-x + y \leq 0$

$$L(x, y, \mu_1, \mu_2) = -(x-2)^2 - 2(y-1)^2 + \mu_1(3 - x - 4y) + \mu_2(0 + x - y)$$

which gives optimality conditions

$$\begin{gathered}
-2(x-2) - \mu_1 + \mu_2 = 0 \\
-4(y-1) - 4\mu_1 - \mu_2 = 0 \\
\mu_1(3 - x - 4y) = 0 \\
\mu_2(x - y) = 0 \\
x + 4y \leq 3 \\
-x + y \leq 0 \\
\mu_1, \mu_2 \geq 0
\end{gathered}$$

Since there are two complementarity conditions, there are four cases to check:
$\mu_1 = 0, \mu_2 = 0$: gives $x = 2$, $y = 1$ which is not feasible.
$\mu_1 = 0, x - y = 0$: gives $x = 4/3, y = 4/3, \mu_2 = -4/3$ which is not feasible.
$\mu_2 = 0, 3 - x - 4y = 0$ gives $x = 5/3, y = 1/3, \mu_1 = 2/3$ which is O.K.
$3 - x - 4y = 0, x - y = 0$ gives $x = 3/5, y = 3/5, \mu_1 = 22/25, \mu_2 = -48/25$ which is not feasible.

Since it is clear that there is an optimal solution, $x = 5/3, y = 1/3$ is it!

Economic Interpretation

The economic interpretation is essentially the same as in the equality case. If the right hand side of a constraint is changed by a small amount Δ, then the optimal objective changes by $\mu^*\Delta$, where μ^* is the optimal Lagrange multiplier corresponding to that constraint. Note that if the constraint is not tight then the objective does not change (since then $\mu^* = 0$).

Handling Nonnegativity

A special type of constraint is nonnegativity. If you have a constraint $x_k \geq 0$, you can write this as $-x_k \leq 0$ and use the above result. This constraint would get a Lagrange multiplier of its own, and would be treated just like every other constraint.

An alternative is to treat nonnegativity implicitly. If x_k must be nonnegative:

1. Change the equality associated with its partial to a less than or equal to zero:

$$\frac{\partial f(x)}{\partial x_k} - \sum_i \lambda_i \frac{\partial g_i(x)}{\partial x_k} - \sum_j \mu_j \frac{\partial h_j(x)}{\partial x_k} \leq 0$$

2. Add a new complementarity constraint:

$$\left(\frac{\partial f(x)}{\partial x_k} - \sum_i \lambda_i \frac{\partial g_i(x)}{\partial x_k} - \sum_j \mu_j \frac{\partial h_j(x)}{\partial x_k} \right) x_k = 0$$

3. Don't forget that $x_k \geq 0$ for x to be feasible.

Sufficiency of conditions

The Karush–Kuhn–Tucker conditions give us candidate optimal solutions x^*. When are these conditions sufficient for optimality? That is, given x^* with λ^* and μ^* satisfying the KKT conditions, when can we be certain that it is an optimal solution?

The most general condition available is:

1. $f(x)$ is concave, and
2. the feasible region forms a convex region.

While it is straightforward to determine if the objective is concave by computing its Hessian matrix, it is not so easy to tell if the feasible region is convex. A useful condition is as follows:

The feasible region is convex if all of the $g_i(x)$ are linear and all of the $h_j(x)$ are convex.

If this condition is satisfied, then any point that satisfies the KKT conditions gives a point that maximizes $f(x)$ subject to the constraints.

2.8 Quadratic Programming and Sequential Quadratic Programming

2.8.1 Quadratic Programming

Quadratic programming is the special case of nonlinear programming

$$\begin{array}{l}\text{Maximize } f(x)\\ \text{Subject to}\\ \quad g_1(x) = b_1\\ \quad\quad\vdots\\ \quad g_m(x) = b_m\\ \quad h_1(x) \leq d_1\\ \quad\quad\vdots\\ \quad h_p(x) \leq d_p\end{array}$$

where the function f is quadratic and the constraints g_i and h_j are linear. The Karush-Khun-Tucker conditions for a quadratic program are linear equations plus nonnegativity on some (or all) the variables and complementarity conditions. For example, the KKT-conditions for

$$\begin{array}{rcl} \textit{minimize } f(x_1, x_2) & = & (x_1 - \frac{1}{2})^2 + (x_2 - \frac{5}{2})^2 \\ & & x_1 - x_2 \geq 0 \\ & & x_1 \geq 0 \\ & & 0 \leq x_2 \leq 2. \end{array}$$

are as follows:

$$\begin{aligned}
2(x_1 - \tfrac{1}{2}) + \mu_1 &= 0 \\
2(x_2 - \tfrac{5}{2}) - \mu_1 - \mu_2 &= 0 \\
-x_1 + x_2 + s_1 &= 0 \\
x_2 + s_2 &= 0, \text{ and} \\
\mu_1 s_1 &= 0 \text{ (Complementarity condition)} \\
\mu_2 s_2 &= 0 \text{ (Complementarity condition)} \\
\mu_1, \mu_2, s_1, s_2 &\geq 0
\end{aligned}$$

Except for the complementarity conditions, this looks very much like linear programming. Indeed, there is a simplex-like algorithm due to Wolfe for solving quadratic programs. The interior point method for linear programming can also be extended efficiently to solve quadratic programs. When the objective function is of the form $f(x) = x^T \Sigma x + Cx$, where Σ is positive definite, then a global optimum can be found. This is the case, for instance, when Σ is a variance-covariance matrix. Very efficient algorithms are available to solve this class of problems.

2.8.2 Sequential Quadratic Programming

To solve a general nonlinear program (NLP):

Maximize $f(x)$
Subject to
$g_1(x) = b_1$
$\vdots$
$g_m(x) = b_m$
$h_1(x) \leq d_1$
$\vdots$
$h_p(x) \leq d_p$

one might try to capitalize on the good algorithms available for solving quadratic programs. This is the idea behind *sequential quadratic programming*. At the current feasible point x^k, the problem (NLP) is approximated by a quadratic program: a quadratic approximation of the objective is computed as well as linear approximations of the equality constraints and of the *active* inequality constraints. The resulting quadratic program is of the form

$$\begin{aligned}
\textit{minimize} \quad & r^k(x - x^k) + \tfrac{1}{2}(x - x^k)^T B_k (x - x^k) \\
& \nabla g_i(x^k)^T (x - x^k) + g_i(x^k) = 0 \quad \text{for all } i \\
& \nabla h_j(x^k)^T (x - x^k) + h_j(x^k) \leq 0 \quad \text{for all active } j
\end{aligned}$$

and can be solved with one of the specialized algorithms. The optimal solution x^{k+1} of the quadratic program is used as the current point for the next iterate. Sequential quadratic programming iterates until the solution converges. A key step is the approximation of (NLP) by a quadratic program, in particular the choice of the vector r^k and matrix B_k in the quadratic approximation of the objective. For details the reader is referred to the survey of Boggs and Tolle in *Acta Numerica 1996*.

2.9 Subgradient Optimization

In this section, we consider unconstrained nonlinear programs of the form

$$\min \quad f(\mathbf{x})$$

where $\mathbf{x} = (x_1, \ldots, x_n)$ and f is a nondifferentiable convex function. Since the gradient is not defined at points x where f is nondifferentiable, the method of steepest descent described in Section 2.5 is not applicable here (remember that this iterative method requires the existence of a gradient direction to make the next step). However the notion of gradient can be generalized as follows. A *subgradient* of f at point x^* is a vector $s^* = (s_1^*, \ldots, s_n^*)$ such that

$$s^*(x - x^*) \leq f(x) - f(x^*) \text{ for every } x.$$

When the function f is differentiable, the subgradient is identical to the gradient. When f is not differentiable at point x, there are typically many subgradients at x. For example, consider the function of one variable

$$f(x) = \max\{1 - x, x - 1\}.$$

This function is nondifferentiable at the point $x = 1$ and it is easy to verify that any vector s such that $-1 \leq s \leq 1$ is a subgradient of f at point $x = 1$.

Consider a nondifferentiable convex function f. The point x^* is a minimum of f if and only if f has a zero subgradient at x^*. In the above example, 0 is a subgradient of f at point $x^* = 1$ and therefore this is where the minimum of f is achieved.

The method of steepest descent can be extended to nondifferentiable convex functions by computing *any* subgradient direction and using the opposite direction to make the next step. Altough subgradient directions are not always directions of ascent, one can nevertheless guarantee convergence to the optimum point by choosing the step size appropriately.

The *subgradient algorithm* can be stated as follows.

1. Initialization: Start from any point x^0. Set $i = 0$.

2. Iteration i: Compute a subgradient s^i of f at point x^i. If s^i is 0 or close to 0, stop. Otherwise, let $x^{i+1} = x^i - d_i s^i$, where $d_i > 0$ denotes a step size, and perform the next iteration.

Several choices of the step size d_i have been proposed in the literature. To guarantee convergence to the optimum, d_i needs to be decreased very slowly. In practice, the following choice is popular: start from $d_0 = 2$ and then half the step size if no improvement in the objective value $f(x^i)$ is observed for k consecutive iterations ($k = 7$ or 8 is often used). The subgradient method is well suited when one wants to get close to the optimum quickly and when finding the exact optimum is not important. With this in mind, a stopping criterion that is frequently used in practice is a maximum number of iterations (say 200) instead of "s^i is 0 or close to 0".

We will see in the next chapter how subgradient optimization is used in a model to construct an index fund.

2.10 Portfolio Optimization

Portfolio optimization is the problem of investing a given capital over a number of available assets in order to maximize the "return" while minimizing the "risk". In the Markowitz model, the "return" of a portfolio is measured by the expected value of the random portfolio return and the "risk" is measured by the variance of the portfolio return. In 1952, Markowitz proposed a quadratic programming model for the construction of *efficient portfolios*, namely portfolios whose expected returns cannot be improved without increasing their variance. In Markowitz's model, future asset prices are unpredictable but information is assumed to be available about their expected returns and covariances. The model minimizes the variance of the portfolio return subject to a given level of expected return and, if so desired, additional investor imposed restrictions on the composition of the portfolio.

Although this technique can, in principle, be used with any variety of financial assets, we will restrict attention to stocks here. For stocks, it is common to compute variances from historical data. Expected returns may be either computed from historical data or estimated from market research data: research analysts assess companies and prepare projections of future performance which can be used in the Markowitz model. When historical data are used to compute variances and/or expected returns, the frequency of the historical data used should depend on the investment horizon of the investor. For short-term investments, daily returns are reasonable.

2.10.1 The Model

Consider n assets. We want to determine the number u_i of units of the i^{th} asset in the portfolio. If short sales are permitted, u_i can be less than 0.

Let x_i denote the weight of the i^{th} asset in the portfolio, namely

$$x_i = \frac{p_i u_i}{p_1 u_1 + p_2 u_2 + \ldots + p_n u_n}$$

where p_i denotes the unit price of the i^{th} asset.

We assume that we have computed the expected yearly return μ_i of each asset as well as the variance σ_i^2 of the returns of each asset and the covariance σ_{ij} between the returns of every pair of assets i and j. To simplify notation, we may also denote the variance by σ_{ii}.

The expected return of the portfolio is then

$$\sum_{i=1}^{n} \mu_i x_i$$

and the variance of the portfolio return is

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_{ij} x_i x_j.$$

In principle, to find an efficient portfolio, we could maximize the expected return on the portfolio for a fixed variance. However, it is computationally easier to handle a nonlinear objective function and a linear constraint than the reverse. Hence, we set up the problem as

the minimization of the portfolio variance subject to a given level of expected return. Fix a desired level of expected return μ. Then the problem is

$$\begin{aligned} & \min \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_{ij} x_i x_j \\ \text{subject to} & \\ & \sum_{i=1}^{n} \mu_i x_i \geq \mu \\ & \sum_{i=1}^{n} x_i = 1 \\ & x_i \text{ unrestricted for } i = 1, \ldots, n \end{aligned}$$

The last statement assumes that short sales are permitted. If short sales are not permitted, then the additional constraints $x_i \geq 0$ for $i = 1, \ldots, n$ need to be added.

This model is thus a quadratic program, with a quadratic objective function and linear constraints. Varying the desired level of expected return μ and plotting the solutions of this problem in mean-variance space traces out the Efficient Frontier.

Estimating the μ_i and σ_{ij}

The Markowitz model gives us an *optimal* portfolio *assuming* that we have perfect information on the μ_i's and σ_{ij}'s for the assets that we are considering. Therefore, an important practical issue is the estimation of the μ_i's and σ_{ij}'s. A reasonable approach for estimating these data is to use time series of past returns (r_{it}= return of asset i from time $t-1$ to time t, where $i = 1, \ldots, n$, $t = 1, \ldots, T$). Unfortunately, it has been observed that small changes in the time series r_{it} lead to changes in the μ_i's and σ_{ij}'s that often lead to significant changes in the "optimal" portfolio.

An attempt to mitigate this problem consists of computing the μ_i's and σ_{ij}'s using a factor model instead of directly from the time series r_{it}. That is, we use the β's of the securities to calculate the μ_i's and σ_{ij}'s, assuming the CAPM holds exactly. This can be done as follows. Let

r_{it}= return of asset i in period t, where $i = 1, \ldots, n$, and $t = 1, \ldots, T$,

r_{mt}= market return in period t,

r_{ft}= return of risk free asset in period t.

Estimate β_i by a standard linear regression

$$r_{it} - r_{ft} = \alpha_i + \beta_i (r_{mt} - r_{ft}) + \epsilon_{it}$$

where the vector ϵ_i represents the idiosyncratic risk of asset i. We assume that $\text{cov}(\epsilon_i, \epsilon_j) = 0$.

Knowing β_i, we compute μ_i by the relation

$$\mu_i - E(r_f) = \beta_i (E(r_m) - E(r_f))$$

and σ_{ij} by the relation

$$\sigma_{ij} = \beta_i \beta_j \sigma_m^2 \quad \text{for } i \neq j$$

$$\sigma_{ii} = \beta_i^2 \sigma_m^2 + \sigma_{\epsilon_i}^2$$

where σ_m^2 denotes the variance of the market return and $\sigma_{\epsilon_i}^2$ the variance of the idiosyncratic risk.

Diversification

In general, there is no reason to expect that solutions to the Markowitz model will be well diversified portfolios. In fact, this model tends to produce portfolios with unreasonably large weights in assets with small capitalization and, when short positions are allowed, unreasonably large short positions. This issue is discussed in Green and Hollifield "When will Efficient Portfolios be Well-Diversified?", *Journal of Finance 1992*. Hence, portfolios chosen by this quadratic program may be subject to idiosyncratic risk. Practitioners often use additional constraints on the x_i's to ensure that the chosen portfolio is well diversified. For example, a limit m may be imposed on the size of each x_i, say

$$x_i \leq m \quad \text{for } i = 1, \ldots, n.$$

One can also reduce sector risk by grouping together investments in securities of a sector and setting a limit on the exposure to this sector. For example, if m_k is the maximum that can be invested in sector k, we add the constraint

$$\sum_{i \text{ in sector } k} x_i \leq m_k.$$

Transaction Costs

We can add a portfolio turnover constraint to ensure that the change between the current holdings x^0 and the desired portfolio x is bounded by h. This constraint is essential in mean-variance models since the covariance matrix is almost singular in most practical applications and hence the optimal decision changes significantly with small changes in the problem data. To avoid big changes when continuously reoptimizing the portfolio, turnover constraints are imposed. Let y_i be the amount of asset i bought and z_i the amount sold. We write

$$x_i - x_i^0 \leq y_i, \quad y_i \geq 0,$$

$$x_i^0 - x_i \leq z_i, \quad z_i \geq 0,$$

$$\sum_{i=1}^{n} (y_i + z_i) \leq h.$$

Instead of a turnover constraint, we can introduce transaction costs directly into the model. Suppose that there is a transaction cost t_i proportional to the amount of asset i bought, and a transaction cost t_i' proportional to the amount of asset i sold. Suppose that the portfolio is reoptimized once per period. As above, let x^0 denote the current portfolio. Then a reoptimized portfolio is obtained by solving

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} \sigma_{ij}x_i x_j$$

subject to

$$\sum_{i=1}^{n}(\mu_i x_i - t_i y_i - t_i' z_i) \geq \mu$$

$$\sum_{i=1}^{n} x_i = 1$$

$$x_i - x_i^0 \leq y_i \quad \text{for } i = 1, \ldots, n$$

$$x_i^0 - x_i \leq z_i \quad \text{for } i = 1, \ldots, n$$

$$y_i \geq 0 \quad \text{for } i = 1, \ldots, n$$

$$z_i \geq 0 \quad \text{for } i = 1, \ldots, n$$

$$x_i \text{ unrestricted for } i = 1, \ldots, n.$$

More on Parameter Estimation

Practitioners have observed that portfolios produced by the Markowitz model are extremely sensitive to the returns μ_i used. Only one small change in one μ_i may produce a totally different portfolio x. What can be done in practice to overcome this problem, or at least reduce it? Michaud (in "Efficient Asset Management: A Practical Guide to Stock Portfolio Management and Asset Allocation" Harvard Business School Press (1998)) recommends to sample the mean returns μ_i and the covariance coefficients σ_{ij} from a confidence interval around each parameter, and then combine the portfolios obtained by solving the Markowitz model for each sample.

Black and Litterman ("Global Portfolio Optimization" *Financial Analysts Journal* 1992) recommend to combine the investor's view with the market equilibrium, as follows.

The expected return vector μ is assumed to have a probability distribution that is the product of two multivariate normal distributions. The first distribution represents the returns at market equilibrium, with mean π and covariance $\tau\Sigma$, where τ is a small constant and $\Sigma = (\sigma_{ij})$ denotes the covariance matrix of asset returns (Note that τ should be small since the variance of the random variable μ is typically much smaller than the variance of the underlying asset returns). The second distribution represents the investor's view about the μ's. These views are expressed as

$$P\mu = q + \epsilon$$

where P is a $k \times n$ matrix and q is a k-dimensional vector that are provided by the investor and ϵ is a normally distributed random vector with mean 0 and diagonal covariance matrix Ω (the stronger the investor's view, the smaller the corresponding ω_i).

The resulting distribution for μ is a multivariate normal distribution with mean

$$\bar{\mu} = [(\tau\Sigma)^{-1} + P^T\Omega^{-1}P]^{-1}[(\tau\Sigma)^{-1}\pi + P^T\Omega^{-1}q].$$

Black and Litterman use $\bar{\mu}$ as the vector of expected returns in the Markowitz model.

Example: Black and Litterman illustrate their approach on a simple three-asset example and then in the context of a seven-country example. Let us go through the three-asset example. Suppose we know the true structure of returns on three assets, A, B and C: For each asset, the return is composed of an equilibrium risk premium plus a common factor and an independent shock.

$$R_A = \pi_A + \gamma_A Z + \nu_A$$
$$R_B = \pi_B + \gamma_B Z + \nu_B$$
$$R_C = \pi_C + \gamma_C Z + \nu_C$$

where

R_i = the return on the ith asset,
π_i = the equilibrium risk premium on the ith asset,
Z = a common factor,
γ_i = the impact of Z on the ith asset,
ν_i = an independent shock to the ith asset.

The covariance matrix Σ of asset returns is assumed to be known. The expected returns of the assets are given by:

$$\mu_i = \pi_i + \gamma_i E[Z] + E[\nu_i].$$

We are not assuming that the world is in equilibrium, i.e. that $E[Z]$ and $E[\nu_i]$ are equal to 0. We do *assume* that the mean μ_i is itself an unobservable random variable whose distribution is centered at the equilibrium risk premium. The uncertainty about μ_i is due to the uncertainty about $E[Z]$ and $E[\nu_i]$. Furthermore we *assume* that the degree of uncertainty about $E[Z]$ and $E[\nu_i]$ is proportional to the volatilities of Z and ν_i respectively. This implies that μ_i is distributed with a covariance structure proportional to Σ. Thus the covariance matrix of expected returns is $\tau\Sigma$ for some scalar τ. Because the uncertainty in the mean is much smaller than the uncertainty in the return itself, τ is close to zero. The equilibrium risk premiums π_i together with $\tau\Sigma$ determine the equilibrium distribution of expected returns. We assume that this information is known to all investors.

In addition, we assume that each individual investor provides additional information about expected returns in terms of views. For example, one type of view is a statement of the form: "I expect that asset A will outperform asset B by 2 %". We interpret such a view to mean that the investor has subjective information about the future returns of assets A and B. We also need a measure of the investor's confidence in his views. This measure is used to determine how much weight to put on the investor's view when combining it with the equilibrium. Consider the limiting case where the investor is 100 % sure of his views. Then we can simply represent the investor's view as a linear restriction on the expected returns:

$$\mu_A - \mu_B = q$$

where here $q = 0.02$. We can then compute the distrubution of $\mu = (\mu_A, \mu_B, \mu_C)^T$ conditional on the equilibrium and this information. This is a relatively straightforward problem in multivariate statistics. To simplify, assume a normal distribution for the means of the random components. The equilibrium distribution of μ is given by the normal distribution $\mathrm{N}(\pi, \tau\Sigma)$ where $\pi = (\pi_A, \pi_B, \pi_C)^T$. To obtain the mean $\bar{\mu}$ of the normal distribution conditional on the linear equation $\mu_A - \mu_B = q$, we need to find the solution to the problem

$\min(\mu - \pi)^T(\tau\Sigma)^{-1}(\mu - \pi)$
subject to $\mu_A - \mu_B = q$.

Let us write the constraint as $P\mu = q$. That is, P is the vector $(1, -1, 0)$. Using the optimality conditions presented in Section 2.7.1, the solution to the above minimization problem can be shown to be

$\bar{\mu} = \pi + (\tau\Sigma)P^T[P(\tau\Sigma)P^T]^{-1}(q - P\pi)$.

For the special case of 100 % confidence in a view, this conditional mean $\bar{\mu}$ is the vector of expected returns that Black and Litterman use in the Markowitz model. In the more general case where the investor is not 100 % confident, they assume that the view can be summarized by a statement of the form $P\mu = q + \epsilon$ where P and q are given by the investor and ϵ is an unobservable normally distributed random variable with mean 0 and variance Ω. When there is more than one view, the vector of views can be represented by $P\mu = q + \epsilon$ where we now interpret P as a matrix (with one row for each view) and ϵ is a normally distributed random vector with mean 0 and diagonal covariance matrix Ω. A diagonal Ω corresponds to the assumption that the views are independent. When this is the case, $\bar{\mu}$ is given by the formula

$\bar{\mu} = [(\tau\Sigma)^{-1} + P^T\Omega^{-1}P]^{-1}[(\tau\Sigma)^{-1}\pi + P^T\Omega^{-1}q]$,

as stated earlier. We refer to the Black and Litterman paper for additional details and an example of an international portfolio.

Mean-Absolute Deviation to Estimate Risk

Konno and Yamazaki ("Mean-Absolute Deviation Portfolio Optimization" *Management Science* 1991) propose a linear programming model instead of the classical quadratic model. Their approach is the following.

The volatility of the portfolio return is

$$\sigma = \sqrt{E[(\sum_{i=1}^{n}(R_i - \mu_i)x_i)^2]}$$

where R_i denotes the random return of asset i.

The L_1-risk of the portfolio return is defined as

$$w = E[|\sum_{i=1}^{n}(R_i - \mu_i)x_i|].$$

Konno and Yamazaki use the following theorem: If $(R_1, \ldots, R_n)$ are multivariate normally distributed random variables, then $w = \sqrt{\frac{2}{\pi}}\sigma$.

This theorem implies that minimizing σ is equivalent to minimizing w when $(R_1, \ldots, R_n)$ is multivariate normally distributed. Then the Markowitz model can be formulated as

$$\min E[|\sum_{i=1}^{n}(R_i - \mu_i)x_i|]$$

subject to

$$\sum_{i=1}^{n} \mu_i x_i \geq \mu$$
$$\sum_{i=1}^{n} x_i = 1$$
$$0 \leq x_i \leq m_i \quad \text{for } i = 1, \ldots, n.$$

Let r_{it} be the realization of random variable R_i during period t for $t = 1, \ldots, T$, which we assume to be available through the historical data or from future projection. Then

$$\mu_i = \frac{1}{T} \sum_{t=1}^{T} r_{it}$$

Furthermore

$$E[|\sum_{i=1}^{n} (R_i - \mu_i) x_i|] = \frac{1}{T} \sum_{t=1}^{T} |\sum_{i=1}^{n} (r_{it} - \mu_i) x_i|$$

Note that the abolute value in this expression makes it nonlinear. But it can be linearized using additional variables. Indeed, one can replace $|x|$ by $y + z$ where $x = y - z$ and $y, z \geq 0$. When the objective is to minimize $y + z$, at most one of y or z will be positive. Therefore the model can be rewritten as

$$\min \sum_{t=1}^{T} y_t + z_t$$

subject to

$$y_t - z_t = \sum_{i=1}^{n} (r_{it} - \mu_i) x_i \quad \text{for } t = 1, \ldots, T$$
$$\sum_{i=1}^{n} \mu_i x_i \geq \mu$$
$$\sum_{i=1}^{n} x_i = 1$$
$$0 \leq x_i \leq m_i \quad \text{for } i = 1, \ldots, n$$
$$y_t \geq 0, \; z_t \geq 0 \quad \text{for } t = 1, \ldots, T$$

This is a linear program! Therefore this approach can be used to solve large scale portfolio optimization problems.

2.10.2 Suggestions for a Project

Investigate the performance of one of the following models: classical Markowitz model, or variations proposed by Michaud, or Black-Litterman or Konno-Yamazaki.

- Choose 30 stocks and retrieve their historical returns over a meaningful horizon.

- Use the historical information to compute expected returns and the variance-covariance matrix for these stock returns.
- Set up the model and solve it with MATLAB or Excel's Solver for different levels μ of expected return. Allow for short sales and include no diversification constraints.
- Recompute these portfolios with no short sales and various diversification constraints.
- Investigate how sensitive the optimal portfolios that you obtained are to small changes in the data (for example how sensitive are they to a small change in the expected return of the assets).
- You currently own the following portfolio: $x_i^0 = 0.20$ for $i = 1, \ldots, 5$ and $x_i^0 = 0$ for $i = 6, \ldots, 30$. Include turnover constraints to reoptimize the portfolio for a fixed level μ of expected return and three different values of h.
- You currently own the following portfolio: $x_i^0 = 0.20$ for $i = 1, \ldots, 5$ and $x_i^0 = 0$ for $i = 6, \ldots, 30$. Reoptimize the portfolio considering transaction costs for buying and selling. Solve for a fixed level μ of expected return and three different transaction costs (0.2 %, 0.5 % and 2 %).

2.11 Value at Risk and Bond Portfolio Optimization

The traditional approach to controling risk has been to balance the mean and variance of returns. In this approach, the efficient frontier is the set of Pareto optimal points with two conflicting criteria: mean and variance. This is appropriate when the portfolio returns are normally distributed. However, when the distibution of returns is heavily skewed and there is a non negligible possibility of large losses, a different measure of risk needs to be used. For example, a portfolio of bonds from emerging markets (Brazil, Russia, Malaysia etc) might be characterized by a large likelihood of small earnings, coupled with a small chance of loosing a large amount of the investment. The loss distribution is heavily skewed and, in this case, standard mean-variance analysis to characterize market risk is inadequate. In this section, we describe a new approach for minimizing *portfolio credit risk*. Credit risk is the risk of a trading partner not fulfilling their obligation in full on the due date or at any time thereafter. Losses can result both from default and from a decline in market value stemming from downgrades in credit ratings. A good reference is the paper of Anderson, Mausser, Rosen and Uryasev, "Credit risk optimization with Conditional Value-at-Risk criterion", *Mathematical Programming B 89* (2001) 273-291.

Let y denote a vector of random events and let $p(y)$ be its known probability distribution. Let x denote a vector of decisions.

1. We make decision x,
2. Random event y is realized.

Let $f(x, y)$ denote the loss function when decision x is made and random event y occurs.

The *Value-at-Risk* (VaR) for a given $0 \leq \beta \leq 1$ is the minimum value of α such that

$$Pr[f(x,y) \leq \alpha] > \beta.$$

It follows from the definition that the VaR α depends on x and β. We denote it by $\alpha_\beta(x)$.

The *Conditional Value-at-Risk* (CVaR) for a given $0 \leq \beta \leq 1$ is the expected loss, conditional on this loss being greater than or equal to the VaR:

$$\Phi_\beta(x) = \frac{1}{1-\beta} \sum_{j:f(x,y_j)\geq\alpha_\beta(x)} p_j f(x,y_j)$$

Example:

Suppose we are given the loss function $f(x,y)$ for a given decision x as $f(x,y) = -y$ where $y = 75 - j$ with probability 1 % for $j = 0,\ldots,99$. We would like to determine the maximum loss incured with 95% probability. This is the value at risk $\alpha_\beta(x)$. Let $\beta = 95\%$. Then the VaR is $\alpha_{95\%}(x) = 20$ since the loss is at most 20 with probability greater than 95 %.

The CVaR is $\Phi_{95\%}(x) = \frac{1}{0.05}(20+21+22+23+24) \times 1\% = 22$.

Model:

Let us focus on CVaR. The basic problem that we would like to solve is the following, where X denotes the set of feasible decisions:

(1) $\min \Phi_\beta(x)$
subject to $x \in X$.

This is not a standard mathematical program since $\alpha_\beta(x)$, which is unknown, appears as a bound in the summation that defines $\Phi_\beta(x)$. To overcome this difficulty, we rewrite $\Phi_\beta(x)$ as follows.

Let $F_\beta(x,\alpha) = \alpha + \frac{1}{1-\beta}\sum_{j=1}^m p_j \max[0, f(x,y_j) - \alpha]$

Then

$\min \Phi_\beta(x) =$ $\min F_\beta(x,\alpha)$
$x \in X$ over $x \in X$ and α

Exercise 7 *Let β and x be given.*
(i) Show that the minimum of $F_\beta(x,\alpha)$ taken over α is $F_\beta(x,\alpha_\beta(x))$.
(ii) Show that $\Phi_\beta(x) = F_\beta(x,\alpha_\beta(x))$.

Thus, to solve (1), it suffices to solve

(2) $\min F_\beta(x,\alpha)$
over $x \in X$ and α.

If $f(x,y_j)$ is a convex function of x, then $F_\beta(x,\alpha)$ is a convex function. Therefore, if X is a convex set, then (2) is a convex nonlinear program. Solving this problem yields the global optimum decision x^*, its VaR α^* and its CVaR $\Phi_\beta(x^*)$.

If $f(x,y_j)$ is a linear function of x and X is given by linear equalities and inequalities, then (2) is a linear program:

$$\begin{array}{rll} \min & \alpha + \frac{1}{1-\beta}\sum_{j=1}^{m} p_j z_j & \\ \text{subject to} & x \in X & \\ & z_j \geq f(x, y_j) - \alpha & \text{for } j = 1, \ldots, m \\ & z_j \geq 0 & \text{for } j = 1, \ldots, m. \end{array}$$

Example: Anderson, Mausser, Rosen and Uryasev consider a portfolio of 197 bonds from 29 different countries with a market value of $ 8.8 billion and duration of approximately 5 years. Their goal is to rebalance the portfolio in order to minimize credit risk. That is they want to minimize losses resulting from default and from a decline in market value stemming from downgrades in credit ratings (credit migration). The loss due to credit migration is simply

$$f(x, y) = (b - y)^T x$$

where b are the future values of each bond with no credit migration and y are the future values with credit migration (so y is a random vector). The one-year portfolio credit loss was generated using a Monte Carlo simulation: 20,000 scenarios of joint credit states of obligators and related losses. The distribution of portfolio losses has a long fat tail. The authors rebalanced the portfolio by minimizing CVaR. The set X of feasible porfolios was described by the following constraints. Let x_i denote the weight of asset i in the portfolio. Upper and lower bounds were set on each x_i:

$$\begin{array}{l} l_i \leq x_i \leq u_i \quad i = 1, \ldots, n \\ \sum_i x_i = 1 \end{array}$$

To calculate the efficient frontier, the expected portfolio return was set to at least μ:

$$\sum_i \mu_i x_i \geq \mu$$

To summarize, the linear program to be solved was as follows:

$$\begin{array}{rll} \min & \alpha + \frac{1}{1-\beta}\sum_{j=1}^{m} p_j z_j & \\ \text{subject to} & z_j \geq \sum_i (b_i - y_{ij}) x_i - \alpha & \text{for } j = 1, \ldots, m \\ & z_j \geq 0 & \text{for } j = 1, \ldots, m \\ & l_i \leq x_i \leq u_i & i = 1, \ldots, n \\ & \sum_i x_i = 1 & \\ & \sum_i \mu_i x_i \geq \mu & \end{array}$$

Consider $\beta = 99\%$. The original bond portfolio had an expected portfolio return of 7.26%. The expected loss was 95 million dollars with a standard deviation of 232 million. The VaR was 1.03 billion dollars and the CVaR was 1.32 billion.

After optimizing the portfolio (with expected return of 7.26%), the expected loss was only 5 thousand dollars, with a standard deviation of 152 million. The VaR was reduced to 210 million and the CVaR to 263 million dollars. So all around, the characteristics of the portfolio were much improved. Positions were reduced in bonds from Brazil, Russia and Venezuela, whereas positions were increased in bonds from Thailand, Malaysia and Chile. Positions in bonds from Colombia, Poland and Mexico remained high and each accounted for about 5 % of the optimized CVaR.

Chapter 3

Integer Programming and Constructing an Index Fund

3.1 Introduction

Consider investing in stocks. A linear programming model might come up with an investment plan that involves buying 3,205.7 shares of stock XYZ. Most people would have no trouble stating that the model suggests bying 3,205 shares or even 3,200 shares. On the other hand, suppose you were trying to find the best among k alternatives (say k job offers). A model that suggests $\frac{1}{k}$ of each would be of little value. A 0,1 decision has to be made, and we would like the model to reflect this.

This integrality restriction may seem rather innocuous, but in reality it has far reaching effects. On one hand, modeling with integer variables has turned out to be useful in a wide variety of applications. With integer variables, one can model logical requirements, fixed costs and many other problem aspects. SOLVER can change a linear programming problem into an integer program with a single command.

The downside of this power, however, is that problems with as few as 40 variables can be beyond the abilities of even the most sophisticated computers. While these small problems are somewhat artificial, real problems with more than 100 or so variables are often not possible to solve unless they show specific exploitable structure. Despite the possibility (or even likelihood) of enormous computing times, there are methods that can be applied to solving integer programs. SOLVER is based on a method called "branch and bound". More sophisticated commercial codes (CPLEX and XPRESS are currently two of the best) use both "branch and bound" and another complementary approach called "cutting plane". The purpose of this chapter is to show some integer programming applications and to describe some of the solution techniques as well as possible pitfalls. The last section of this chapter discusses a problem in finance that can be modeled as an integer program: constructing an index fund. Another example will be given in the next chapter: structuring collaterized mortgage obligations (CMO's).

First we introduce some terminology. An integer programming problem in which all variables are required to be integer is called a *pure integer programming problem*. If some variables are restricted to be integer and some are not then the problem is a *mixed integer*

programming problem. The case where the integer variables are restricted to be 0 or 1 comes up surprising often. Such problems are called *pure (mixed) 0 1 programming problems* or *pure (mixed) binary integer programming problems*.

3.2 Modeling with Integer Variables

3.2.1 Capital Budgeting

Suppose we wish to invest \$19,000. We have identified four investment opportunities. Investment 1 requires an investment of \$6,700 and has a net present value of \$8,000; investment 2 requires \$10,000 and has a value of \$11,000; investment 3 requires \$5,500 and has a value of \$6,000; and investment 4 requires \$3,400 and has a value of \$4,000. Into which investments should we place our money so as to maximize our total present value? Each project is a "take it or leave it" opportunity: it is not allowed to invest partially in any of the projects.

As in linear programming, our first step is to decide on our variables. This can be much more difficult in integer programming because there are very clever ways to use integrality restrictions. In this case, we will use a 0–1 variable x_j for each investment. If x_j is 1 then we will make investment j. If it is 0, we will not make the investment.

This leads to the 0–1 programming problem:

Maximize $8x_1 + 11x_2 + 6x_3 + 4x_4$
subject to
$$6.7x_1 + 10x_2 + 5.5x_3 + 3.4x_4 \leq 19$$
$x_j = 0$ or 1.

Now, a straightforward "bang for buck" suggests that investment 1 is the best choice. In fact, ignoring integrality constraints, the optimal linear programming solution is $x_1 = 1, x_2 = 0.89, x_3 = 0, x_4 = 1$ for a value of \$21,790. Unfortunately, this solution is not integral. Rounding x_2 down to 0 gives a feasible solution with a value of \$12,000. There is a better integer solution, however, of $x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 1$ for a value of \$21,000. This example shows that rounding does not necessarily give an optimal solution.

There are a number of additional constraints we might want to add. For instance, consider the following constraints:

1. We can only make two investments.
2. If investment 2 is made, then investment 4 must also be made.
3. If investment 1 is made, then investment 3 cannot be made.

All of these, and many more *logical restrictions*, can be enforced using 0–1 variables. In these cases, the constraints are:

1. $x_1 + x_2 + x_3 + x_4 \leq 2$
2. $x_2 - x_4 \leq 0$
3. $x_1 + x_3 \leq 1$.

Solving the model with SOLVER

Modeling an integer program in SOLVER is almost the same as modeling a linear program. For example, if you placed binary variables x_1, x_2, x_3, x_4 in cells \$B\$5:\$B\$8, simply `Add` the constraint

```
$B$5:$B$8 Bin
```

to your other constraints in the SOLVER dialog box. Note that the `Bin` option is found in the small box where you usually indicate the type of inequality: `=`, `<=` or `>=`. Just click on `Bin`. That's all there is to it!

It is equally easy to model an integer program within other commercial codes. The formulation might look as follows.

```
   !  Capital budgeting example
VARIABLES
x(i=1:4)
OBJECTIVE
Max:  8*x(1) + 11*x(2) + 6*x(3) + 4*x(4)
CONSTRAINTS
Budget:  6.7*x(1) + 10*x(2) + 5.5*x(3) + 3.4*x(4) < 19
BOUNDS
x(i=1:4) Binary
END
```

Exercise 8 (Optional) *As the leader of an oil exploration drilling venture, you must determine the best selection of 5 out of 10 possible sites. Label the sites $s_1, s_2, \ldots, s_{10}$ and the expected profits associated with each as $p_1, p_2, \ldots, p_{10}$.*

(i) If site s_2 is explored, then site s_3 must also be explored. Furthermore, regional development restrictions are such that

(ii) Exploring sites s_1 and *s_7 will prevent you from exploring site s_8.*

(iii) Exploring sites s_3 or *s_4 will prevent you from exploring site s_5.*

Formulate an integer program to determine the best exploration scheme and solve with SOLVER.

3.2.2 The Lockbox Problem

Consider a national firm that receives checks from all over the United States. Due to the vagaries of the U.S. Postal Service, as well as the banking system, there is a variable delay from when the check is postmarked (and hence the customer has met her obligation) and when the check clears (and when the firm can use the money). For instance, a check mailed in Pittsburgh sent to a Pittsburgh address might clear in just 2 days. A similar check sent to Los Angeles might take 4 days to clear. It is in the firm's interest to have the check clear as quickly as possible since then the firm can use the money. In order to speed up this clearing, firms open offices (called lockboxes) in different cities to handle the checks.

For example, suppose we receive payments from 4 regions (West, Midwest, East, and South). The average daily value from each region is as follows: \$300,000 from the West, \$120,000 from the Midwest, \$360,000 from the East, and \$180,000 from the South. We are considering opening lockboxes in L.A., Cincinnati, Boston, and/or Houston. Operating a lockbox costs \$90,000 per year. The average days from mailing to clearing is given in the table. Which lockboxes should we open?

From	L.A.	Cincinnati	Boston	Houston
West	2	4	6	6
Midwest	4	2	5	5
East	6	5	2	5
South	7	5	6	3

Table 3.1: Clearing Times

First we must calculate the losses due to lost interest for each possible assignment. For example, if the West sends to Boston, then on average there will be \$1,800,000 ($= 6 \times \$300,000$) in process on any given day. Assuming an investment rate of 10%, this corresponds to a yearly loss of \$180,000. We can calculate the losses for the other possibilities in a similar fashion to get table 3.2.

From	L.A.	Cincinnati	Boston	Houston
West	60	120	180	180
Midwest	48	24	60	60
East	216	180	72	180
South	126	90	108	54

Table 3.2: Lost Interest ('000)

The formulation takes a bit of thought. Let y_j be a 0–1 variable that is 1 if lockbox j is opened and 0 if it is not. Let x_{ij} be 1 if region i sends to lockbox j.

Our objective is to minimize our total yearly costs. This is:

$60x_{11} + 120x_{12} + 180x_{13} + 180x_{14} + 48x_{21} + \ldots + 90y_1 + 90y_2 + 90y_3 + 90y_4$.

One set of constraints is as follows:

$\sum_j x_{ij} = 1$ for all i (each region must be assigned to one lockbox).

A more difficult set of constraints is that a region can only be assigned to an open lockbox. For lockbox 1 (L.A.), this can be written

$$x_{11} + x_{21} + x_{31} + x_{41} \leq 100y_1$$

(There is nothing special about 100; any number at least 4 would do.) Suppose we do not open L.A. Then y_1 is 0, so all of x_{11}, x_{21}, x_{31}, and x_{41} must also be. If y_1 is 1 then there is no restriction on the x values.

We can create constraints for the other lockboxes to finish off the integer program. For this problem, we would have 20 variables (4 y variables, 16 x variables) and 8 constraints. This gives the following integer program:

```
MIN     60 X11 + 120 X12 + 180 X13 + 180 X14 + 48 X21
      + 24 X22 + 60 X23 + 60 X24 + 216 X31 + 180 X32
      + 72 X33 + 180 X34 + 126 X41 + 90 X42 + 108 X43
      + 54 X44 + 90 Y1 + 90 Y2 + 90 Y3 + 90 Y4
SUBJECT TO
           X11 + X12 + X13 + X14 =    1
           X21 + X22 + X23 + X24 =    1
           X31 + X32 + X33 + X34 =    1
           X41 + X42 + X43 + X44 =    1
           X11 + X21 + X31 + X41 - 100 Y1 <=   0
           X12 + X22 + X32 + X42 - 100 Y2 <=   0
           X13 + X23 + X33 + X43 - 100 Y3 <=   0
           X14 + X24 + X34 + X44 - 100 Y4 <=   0
ALL VARIABLES BINARY
```

If we ignore integrality, we get the solution $x_{11} = x_{22} = x_{33} = x_{44} = 1$, $y_1 = y_2 = y_3 = y_4 = 0.01$ and the rest equals 0. Note that we get no useful information out of this linear programming solution.

The above is a perfectly reasonable 0–1 programming formulation of the lockbox problem. Note that many variations are possible (Boston costs more to operate in than other cities, South won't send to L.A., and so on).

There are other formulations, however. For instance, consider the sixteen constraints of the form

$$x_{ij} \leq y_j$$

These constraints also force a region to only use open lockboxes (check this!). It might seem that a larger formulation is less efficient and therefore should be avoided. This is not the case! If we solve the linear program with the above constraints, we get the solution $x_{11} = x_{21} = x_{33} = x_{43} = y_1 = y_3 = 1$ with the rest equal to zero. In fact, we have an integer solution, which must therefore be optimal! Different formulations can have very different properties with respect to their associated linear program. One very active research area is to take common problems and find good reformulations.

The above is an example of a *fixed charge* problem. There is a fixed charge for opening a lockbox, but then it can be used as much as desired. There are many other types of fixed charge problems.

3.2.3 Set Covering

To illustrate this model, consider the following *location* problem: A city is reviewing the location of its fire stations. The city is made up of a number of neighborhoods, as illustrated in figure 3.1.

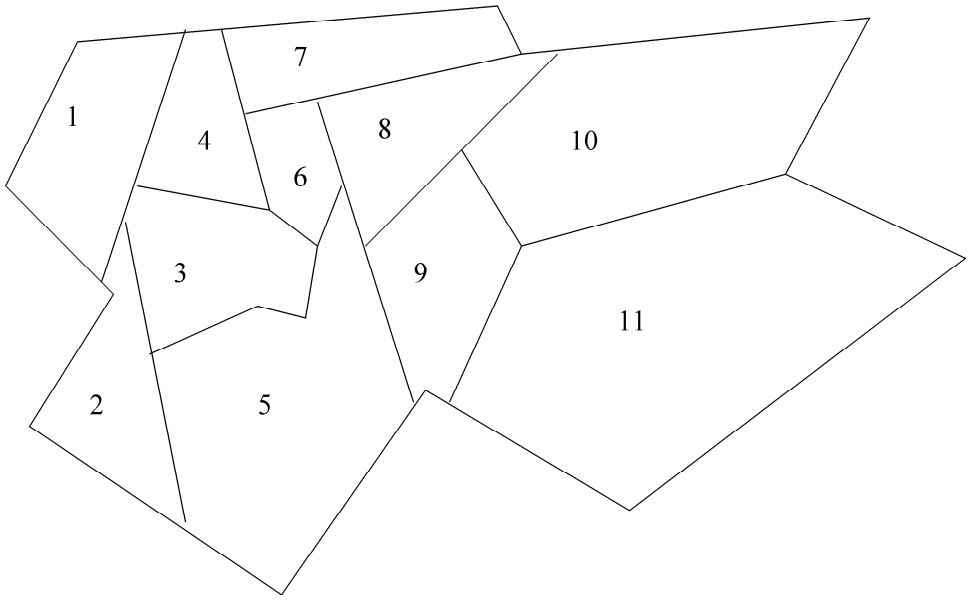

Figure 3.1: Map of the city

A fire station can be placed in any neighborhood. It is able to handle the fires for both its neighborhood and any adjacent neighborhood (any neighborhood with a non–zero border with its home neighborhood). The objective is to minimize the number of fire stations used.

We can create one variable x_j for each neighborhood j. This variable will be 1 if we place a station in the neighborhood, and will be zero otherwise. This leads to the following formulation

```
MIN   X1 + X2 + X3 + X4 + X5 + X6 + X7 + X8 + X9 +X10 +X11
SUBJECT TO
       1)   X1 + X2 + X3 + X4 >=   1
       2)   X1 + X2 + X3 + X5 >=   1
       3)   X1 + X2 + X3 + X4 + X5 + X6 >=   1
       4)   X1 + X3 + X4 + X6 + X7 >=   1
       5)   X2 + X3 + X5 + X6 + X8 + X9 >=   1
       6)   X3 + X4 + X5 + X6 + X7 + X8 >=   1
       7)   X4 + X6 + X7 + X8 >=   1
       8)   X5 + X6 + X7 + X8 + X9 + X10 >=   1
      19)   X5 + X8 + X9 + X10 + X11 >=   1
      10)   X8 + X9 + X10 + X11 >=   1
      11)   X9 + X10 + X11 >=   1
ALL VARIABLES BINARY
```

The first constraint states that there must be a station either in neighborhood 1 or in some adjacent neighborhood. The next constraint is for neighborhood 2 and so on.

One optimal solution to this is $x_3 = 1, x_8 = 1, x_9 = 1$ and the rest have value 0.

This is an example of the *set covering problem*. The set covering problem is characterized by having binary variables, $\geq$ constraints each with a right hand side of 1, and having simply sums of variables as constraints. In general, the objective function can have any coefficients, though here it is of a particularly simple form.

3.2.4 Traveling Salesperson Problem

Consider a traveling salesperson who must visit each of 20 cities before returning home. She knows the distance between each of the cities and wishes to minimize the total distance traveled while visiting all of the cities. In what order should she visit the cities?

Let there be n cities, numbered from 1 up to n. For each pair of cities (i, j) let c_{ij} be the cost of going from city i to city j. Let's let x_{ij} be 1 if the person travels between cities i and j (either from city i to city j or from j to i).

The objective is to minimize $\sum_{i=1}^{n} \sum_{j=1}^{i-1} c_{ij} x_{ij}$. The constraints are harder to find. Consider the following set:

$$\sum_{j \neq i} x_{ij} = 2 \text{ for all } i.$$

These constraints say that every city must be visited. These constraints are not enough, however, since it is possible to have multiple cycles, rather than one big cycle through all the points. To handle this, we can use the following set of constraints:

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 2 \text{ for all } S \subset N.$$

This set states that, for any subset of cities S, the tour must enter and exit that set. These, together with $x_{ij} \in \{0, 1\}$ is sufficient to formulate the traveling salesperson problem (TSP) as an integer program.

Note however, that there are a tremendous number of constraints: for our 20 city problem, that number is roughly 524,288. For a 300 city problem, this would amount to 10185179881672 430431342228442046890805257341968329681253180702246771906498816683530916986 88 constraints... Try putting that into SOLVER!

Despite the apparent complexity of this formulation, it lies at the heart of the most promising current approach to solving medium (100–1000 city) TSPs. We will see this again in the cutting plane section.

3.3 Solving Integer Programs

We have gone through a number of examples of integer programs. A natural question is "How can we get solutions to these models?" There are two common approaches. Historically, the first method developed was based on cutting planes (adding constraints to force integrality). In the last forty years or so, however, the most effective technique has been based on dividing the problem into a number of smaller problems in a method called branch and bound. Recently (the last ten years or so), cutting planes have made a resurgence in the form of facets and polyhedral characterizations. All these approaches involve solving a series of *linear* programs. So that is where we will begin.

3.3.1 Relationship to Linear Programming

Given an integer program

$$\begin{aligned}\text{(IP) Min (or Max) } & cx \\ & Ax = b \\ & x \geq 0 \\ & x \text{ integer.}\end{aligned}$$

there is an associated linear program called the *linear relaxation* formed by dropping the integrality restrictions:

$$\begin{aligned}\text{(LR) Min (or Max) } & cx \\ & Ax = b \\ & x \geq 0.\end{aligned}$$

Since LR is less constrained than IP, the following are immediate:

- If IP is a minimization, the optimal objective value for LR is less than or equal to the optimal objective for IP.
- If IP is a maximization, the optimal objective value for LR is greater than or equal to that of IP.
- If LR is infeasible, then so is IP.
- If LR is optimized by integer variables, then that solution is feasible and optimal for IP.
- If the objective function coefficients are integer, then, for minimization, the optimal objective for IP is greater than or equal to the "round up" of the optimal objective for LR. For maximization, the optimal objective for IP is less than or equal to the "round down" of the optimal objective for LR.

So solving LR does give some information: it gives a bound on the optimal value, and, if we are lucky, may give the optimal solution to IP. We saw, however, that rounding the solution of LR will not in general give the optimal solution of IP. In fact, for some problems it is difficult to round and even get a feasible solution.

Exercise 9 (Optional) *Consider the problem*

$$\begin{aligned}Max\ & 20x_1 + 10x_2 + 10x_3 \\ & 2x_1 + 20x_2 + 4x_3 \leq 15 \\ & 6x_1 + 20x_2 + 4x_3 = 20 \\ & x_1, x_2, x_3 \geq 0 \text{ integer.}\end{aligned}$$

Solve this problem as a linear program. Then, show that it is impossible to obtain a feasible integer solution by rounding the values of the variables.

3.3.2 Branch and Bound

We will explain branch and bound by solving the following integer program.

Max $8x_1 + 11x_2 + 6x_3 + 4x_4$
$5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14$
$x_j = 0$ or 1.

The linear relaxation solution is $x_1 = 1, x_2 = 1, x_3 = 0.5, x_4 = 0$ with a value of 22. We know that no integer solution will have value more than 22. Unfortunately, since x_3 is not integer, we do not have an integer solution yet.

We want to force x_3 to be integer. To do so, we *branch* on x_3, creating two new problems. In one, we will add the constraint $x_3 = 0$. In the other, we add the constraint $x_3 = 1$.

Note that any optimal solution to the overall problem must be feasible to one of the subproblems. If we solve the linear relaxations of the subproblems, we get the following solutions:

$x_3 = 0$: objective 21.65, $x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 0.667$

$x_3 = 1$: objective 21.85, $x_1 = 1.x_2 = .714, x_3 = 1, x_4 = 0$.

At this point we know that the optimal integer solution is no more than 21.85 (we actually know it is less than or equal to 21 (Why?)), but we still do not have any feasible integer solution. So, we will take a subproblem and branch on one of its variables. In general, we will choose the subproblem as follows:

— We will choose an *active* subproblem, which so far only means one we have not chosen before, and

— We will choose the subproblem with the highest solution value (for maximization) (lowest for minimization).

In this case, we will choose the subproblem with $x_3 = 1$, and branch on x_2. Solving the resulting subproblems, we get:

$x_3 = 1, x_2 = 0$: objective 18, $x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 1$,

$x_3 = 1, x_2 = 1$: objective 21.8, $x_1 = 0.6, x_2 = 1, x_3 = 1, x_4 = 0$.

We now have a feasible integer solution with value 18. Furthermore, since the $x_3 = 1, x_2 = 0$ problem gave an integer solution, no further branching on that problem is necessary. It is not active due to integrality of solution. There are still active subproblems that might give values more than 18. Using our rules, we will branch on problem $x_3 = 1, x_2 = 1$ by branching on x_1 to get:

$x_3 = 1, x_2 = 1, x_1 = 0$: objective 21, $x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 1$,

$x_3 = 1, x_2 = 1, x_1 = 1$: infeasible.

Our best integer solution now has value 21. The subproblem that generates that is not active due to integrality of solution. The other subproblem generated is not active due to infeasibility. There is still a subproblem that is active. It is the subproblem with solution value 21.65. By our "round–down" result, there is no better solution for this subproblem than 21. But we already have a solution with value 21. It is not useful to search for another such solution. We can *fathom* this subproblem based on the above bounding argument and mark it not active. There are no longer any active subproblems, so the optimal solution value is 21.

We have seen all parts of the branch and bound algorithm. The essence of the algorithm is as follows:

1. Solve the linear relaxation of the problem. If the solution is integer, then we are done. Otherwise create two new subproblems by branching on a fractional variable.

2. A subproblem is not active when any of the following occurs:

   (a) You used the subproblem to branch on,

   (b) All variables in the solution are integer,

   (c) The subproblem is infeasible,

   (d) You can fathom the subproblem by a bounding argument.

3. Choose an active subproblem and branch on a fractional variable. Repeat until there are no active subproblems.

That's all there is to branch and bound! Depending on the type of problem, the branching rule may change somewhat. For instance, if x is restricted to be integer (but not necessarily 0 or 1), then if $x = 4.27$ your would branch with the constraints $x \leq 4$ and $x \geq 5$ (not on $x = 4$ and $x = 5$).

In the worst case, the number of subproblems can get huge. For many problems in practice, however, the number of subproblems is quite reasonable.

For an example of a huge number of subproblems, try the following in SOLVER:

```
MAX   - X0 + 2 X1 + 2 X2 + 2 X3 + 2 X4 + 2 X5 + 2 X6 + 2 X7
      + 2 X8 + 2 X9 + 2 X10 + 2 X11 + 2 X12 + 2 X13 + 2 X14
      + 2 X15 + 2 X16 + 2 X17
 SUBJECT TO
        2)   X0 + 2 X1 + 2 X2 + 2 X3 + 2 X4 + 2 X5 + 2 X6
      + 2 X7 + 2 X8 + 2 X9 + 2 X10 + 2 X11 + 2 X12 + 2 X13
      + 2 X14 + 2 X15 + 2 X16 + 2 X17 <=   17
 ALL VARIABLES BINARY
```

Note that this has only 18 variables. SOLVER looks at 48,619 subproblems, before deciding the optimal objective is 16. The 100 variable version of this problem takes about 10^{29} subproblems or about 3×10^{18} years (at 1000 subproblems per second). Luckily, most problems take far less time.

Exercise 10 (Optional) *Solve the following problem by the branch and bound algorithm. For convenience, always select x_1 as the branching variable when both x_1 and x_2 are fractional.*

Max $x_1 + x_2$
$2x_1 + 5x_2 \leq 16$
$6x_1 + 5x_2 \leq 30$
$x_1, x_2 \geq 0$ *integer.*

Exercise 11 (Optional) *Repeat Exercise 10 assuming that x_1 only is restricted to integer values.*

3.3.3 Cutting Plane Techniques

There is an alternative to branch and bound called *cutting planes* which can also be used to solve integer programs. The fundamental idea behind cutting planes is to add constraints to a linear program until the optimal basic feasible solution takes on integer values. Of course, we have to be careful which constraints we add: we would not want to change the problem by adding the constraints. We will add a special type of constraint called a *cut*. A cut relative to a current fractional solution satisfies the following criteria:

1. every feasible integer solution is feasible for the cut, and
2. the current fractional solution is not feasible for the cut.

This is illustrated in figure 3.2.

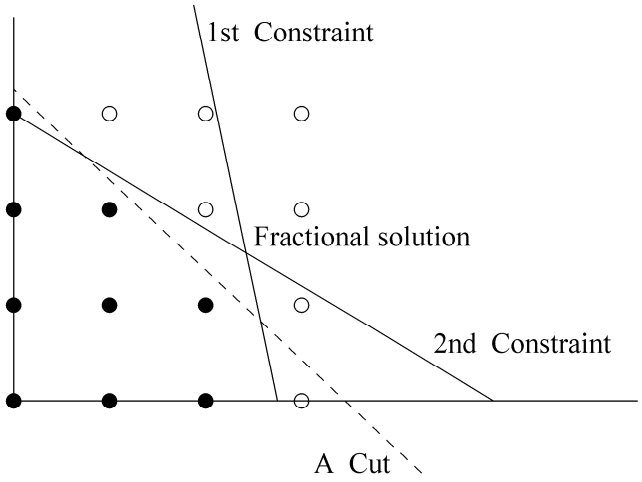

Figure 3.2: A cut

There are two ways to generate cuts. The first, called Gomory cuts, generates cuts from any linear programming tableau. This has the advantage of "solving" any problem but has the disadvantage that the method can be very slow. The second approach is to use the structure of the problem to generate very good cuts. The approach needs a problem–by–problem analysis, but can provide very efficient solution techniques.

General cutting planes

Consider the following integer program:

$$\begin{array}{rrcrcl} Max & 7x_1 & + & 9x_2 & & \\ & -x_1 & + & 3x_2 & \leq & 6 \\ & 7x_1 & + & x_2 & \leq & 35 \\ & x_1, & & x_2 & \geq & 0 \ \textit{integer.} \end{array}$$

If we ignore integrality, we get the following optimal tableau:

Variable	x_1	x_2	s_1	s_2	RHS
z	0	0	28/11	15/11	63
x_2	0	1	7/22	1/22	7/2
x_1	1	0	-1/22	3/22	9/2

Let's look at the first constraint:

$$x_2 + 7/22s_1 + 1/22s_2 = 7/2$$

We can manipulate this to put all of the integer parts on the left side, and all the fractional parts on the right to get:

$$x_2 - 3 = 1/2 - 7/22s_1 - 1/22s_2$$

Now, note that the left hand side consists only of integers, so the right hand side must add up to an integer. Which integer can it be? Well, it consists of some positive fraction minus a series of positive values. Therefore, the right hand side can only be $0, -1, -2, \ldots$, it cannot be a positive value. Therefore, we have derived the following constraint:

$$1/2 - 7/22s_1 - 1/22s_2 \leq 0.$$

This constraint is satisfied by every feasible integer solution to our original problem. But, in our current solution, s_1 and s_2 both equal 0, which is infeasible to the above constraint. This means the above constraint is a cut, called the *Gomory cut* after its discoverer. We can now add this constraint to the linear program and be guaranteed to find a different solution, one that might be integer.

We can also generate a cut from the other constraint. Here we have to be careful to get the signs right:

$$x_1 - 1/22s_1 + 3/22s_2 = 9/2$$
$$x_1 + (-1 + 21/22)s_1 + 3/22s_2 = 4 + 1/2$$
$$x_1 - s_1 - 4 = 1/2 - 21/22s_1 - 3/22s_2$$

gives the constraint

$$1/2 - 21/22s_1 - 3/22s_2 \leq 0.$$

In general, let $\lfloor a \rfloor$ be defined as the largest integer less than or equal to a. This implies $\lfloor 3.9 \rfloor = 3$, $\lfloor 5 \rfloor = 5$, and $\lfloor -1.3 \rfloor = -2$.

If we have a constraint

$$x_k + \sum a_i x_i = b$$

with b not an integer, we can write each $a_i = \lfloor a_i \rfloor + a_i'$, for some $0 \leq a_i' < 1$, and $b = \lfloor b \rfloor + b'$ for some $0 < b' < 1$. Using the same steps we get:

$$x_k + \sum \lfloor a_i \rfloor x_i - \lfloor b \rfloor = b' - \sum a_i' x_i$$

to get the cut

$$b' - \sum a_i' x_i \leq 0.$$

This cut can then be added to the linear program and the problem resolved. The problem is guaranteed not to get the same solution.

This method can be shown to guarantee finding the optimal integer solution. There are a couple of disadvantages:

1. Round–off error can cause great difficulties: Is that 3.000000001 really a 3, or should I generate a cut? If I make the wrong decision I could either cut off a feasible solution (if it is really a 3 but I generate a cut) or I could end up with an infeasible solution (if it is not a 3 but I treat it as one).

2. The number of constraints that are generated can be enormous. Just like branch and bound can generate a huge number of subproblems, this technique can generate a huge number of constraints.

The combination of these makes this cutting plane technique impractical by itself. Recently however, more powerful techniques have been discovered for special problem structure. This is the subject of the next section.

Cuts for special structure

Gomory cuts have the property that they can be generated for any integer program. Their weakness is their downfall: they do not seem to cut off much more than the linear programming solution in practice. An alternative approach is to generate cuts that are specially designed for the particular application. We saw that in a simple form in the lockbox problem, where we used the constraints $x_{ij} \leq y_j$ because they were stronger than $\sum_i x_{ij} \leq 100y_j$. In this section, we examine the traveling salesperson problem.

Recall that there is no good, compact formulation for the TSP. Earlier, we generated a formulation as follows:

Min $\sum_{ij} c_{ij}x_{ij}$
$\sum_{j \neq i} x_{ij} = 2$ for all i,
$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 2$ for all $S \subset N$,
$x_{ij} \in \{0, 1\}$ for all i, j.

Recall that in the second set of constraints (called *subtour elimination* constraints) there are many, many constraints. One approach is to initially ignore these constraints and simply solve the problem over the first set of constraints. Suppose we have a six node problem as shown in figure 3.3.

This problem can be formulated as follows:

```
MIN     4 X12 + 4 X13 + 3 X14 + 4 X23 + 2 X25 + 3 X36
     + 4 X45 + 4 X46 + 4 X56
SUBJECT TO
          X12 + X13 + X14  =    2
          X12 + X23 + X25  =    2
          X13 + X23 + X36  =    2
          X14 + X45 + X46  =    2
          X25 + X45 + X56  =    2
          X36 + X46 + X56  =    2
          X12 <=   1
          X13 <=   1
          X14 <=   1
```

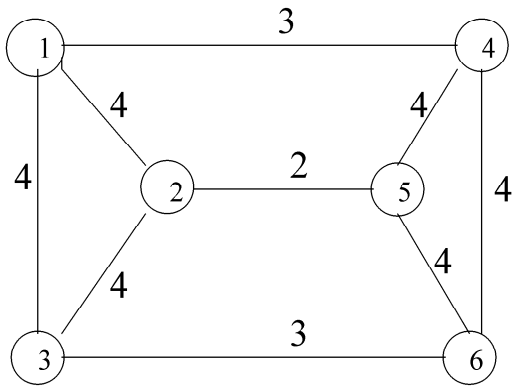

Figure 3.3: Six node traveling salesperson problem

```
          X23 <=   1
          X25 <=   1
          X36 <=   1
          X45 <=   1
          X46 <=   1
          X56 <=   1
END

LP OPTIMUM FOUND AT STEP      2

       OBJECTIVE FUNCTION VALUE

       1)     20.000000

 VARIABLE        VALUE          REDUCED COST
      X12         0.500000          0.000000
      X13         0.500000          0.000000
      X14         1.000000          0.000000
      X23         0.500000          0.000000
      X25         1.000000          0.000000
      X36         1.000000          0.000000
      X45         0.500000          0.000000
      X46         0.500000          0.000000
      X56         0.500000          0.000000
```

This solution, while obviously not a tour, actually satisfies all of the subtour elimination constraints. At this point we have three choices:

1. We can do branch and bound on our current linear program, or

2. We can apply Gomory cuts to the resulting tableau, or

3. We can try to find other classes of cuts to use.

In fact, for the traveling salesperson problem, there are a number of other classes of cuts to use. These cuts look at different sets of arcs and try to say something about how a tour can use them. For instance, look at the set of arcs in figure 3.4:

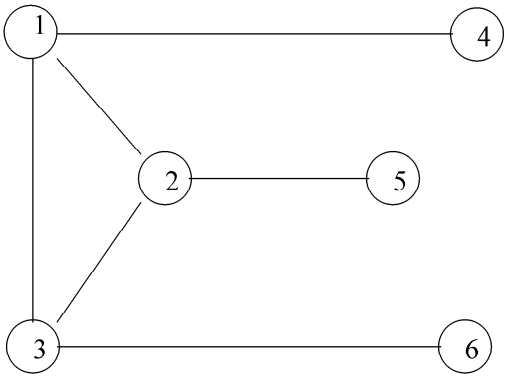

Figure 3.4: Arc Set for Comb Inequality

It is fairly easy to convince yourself that no tour can use more than 4 of these arcs. This is an example of a broad class of inequalities called *comb* inequalities. This means that the inequality stating that the sum of the x values on these arcs is less than or equal to 4 is a valid inequality (it does not remove any feasible solution to the integer program). Our solution, however, has 4.5 units on those arcs. Therefore, we can add a constraint to get the following formulation and result:

```
MIN     4 X12 + 4 X13 + 3 X14 + 4 X23 + 2 X25 + 3 X36
     + 4 X45 + 4 X46 + 4 X56
SUBJECT TO
          X12 + X13 + X14  =    2
          X12 + X23 + X25  =    2
          X13 + X23 + X36  =    2
          X14 + X45 + X46  =    2
          X25 + X45 + X56  =    2
          X36 + X46 + X56  =    2
          X12 + X13 + X23 + X14 + X25 + X36  <=   4
          X12 <=   1
          X13 <=   1
          X14 <=   1
          X23 <=   1
          X25 <=   1
          X36 <=   1
          X45 <=   1
          X46 <=   1
          X56 <=   1
```

```
END

 LP OPTIMUM FOUND AT STEP      2

        OBJECTIVE FUNCTION VALUE

        1)     21.000000

  VARIABLE        VALUE          REDUCED COST
       X12         1.000000          0.000000
       X13         1.000000          0.000000
       X14         0.000000          0.000000
       X23         0.000000          0.000000
       X25         1.000000          0.000000
       X36         1.000000          0.000000
       X45         1.000000          0.000000
       X46         1.000000          0.000000
       X56         0.000000          0.000000
```

So, we have the optimal solution.

This method, perhaps combined with branch and bound if the solution is still fractional after all known inequalities are examined, has proven to be a very practical and robust method for solving medium–sized TSPs. Futhermore, many classes of inequalities are known for other combinatorial optimization problems. This approach, seriously studied only for the last five years or so, has greatly increased the size and type of instances that can be effectively solved to optimality.

Exercise 12 (Optional) *For the problem*

$$\begin{aligned} &Max\ 8x_1 + 11x_2 + 6x_3 + 4x_4 \\ &\quad 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 \\ &\quad x_j = 0 \text{ or } 1, \end{aligned}$$

show that the following inequalities do not remove any feasible solutions:

$$x_1 + x_2 + x_3 \leq 2,$$

$$x_1 + x_2 + x_4 \leq 2.$$

Use these inequalities to solve the problem by SOLVER as a linear program (i.e. the 0-1 restrictions on x_j are replaced by $x_j \leq 1$).

Solutions of Optional Exercises

Exercise 8:

$$
\begin{array}{rrcl}
Max & \sum_{j=1}^{10} p_j x_j & & \\
\text{subject to} & & & \\
& \sum_{j=1}^{10} x_j & = & 5 \\
& x_2 - x_3 & \leq & 0 \\
& x_1 + x_7 + x_8 & \leq & 2 \\
& x_3 + x_5 & \leq & 1 \\
& x_4 + x_5 & \leq & 1 \\
& x_j & = & 0 \text{ or } 1 \text{ for } j = 1, \ldots, 10.
\end{array}
$$

Exercise 9: First we solve the problem as a linear program with SOLVER. The solution is $x_1 = 3.33333$, $x_2 = x_3 = 0$. Rounding it yields $x_1 = 3$, $x_2 = x_3 = 0$ which fails to satisfy the constraint $6x_1 + 20x_2 + 4x_3 = 20$.

In fact, the only feasible integer solution is

$$x_1 = 2, x_2 = 0, x_3 = 2,$$

and it cannot be obtained by simple rounding.

Exercise 10:

Using the branch and bound technique, three optimum integer solutions were found, namely

$$\left\{ \begin{array}{l} x_1 = 3 \\ x_2 = 2 \end{array} \right. \quad \left\{ \begin{array}{l} x_1 = 4 \\ x_2 = 1 \end{array} \right. \quad \left\{ \begin{array}{l} x_1 = 5 \\ x_2 = 0 \end{array} \right.$$

each with value $z = 5$.

Exercise 11:

Using the branch and bound technique, we find that the optimum solution is $x_1 = 4, x_2 = 1.2$ with value $z_3 = 5.2$.

Exercise 12:

Since $x_1 = x_2 = x_3 = 1$ does not satisfy the constraint $5x_1 + 7x_2 + 4x_3 + x_4 \leq 14$, it follows that $x_1 + x_2 + x_3 \leq 2$ must hold for all solutions that satisfy the constraint.

Similarly for $x_1 + x_2 + x_4 \leq 2$.

```
MAX 8*X(1) + 11*X(2) + 6*X(3) + 4*X(4)
SUBJECT TO
Budget: 5*X(1) + 7*X(2) + 4*X(3) + 3*X(4)  < 14
Cut1: X(1) +  X(2) +  X(3) < 2
Cut2: X(1) +  X(2) +  X(4) < 2
Bounds: X(i=1:4) < 1
```

The linear programming solution is $x_1 = 0$, $x_2 = x_3 = x_4 = 1$ with objective value 21.

3.4 Constructing an Index Fund

An old and recurring debate about investing lies in the merits of active versus passive management of a portfolio. Active portfolio management tries to achieve superior performance by using technical and fundamental analysis as well as forecasting techniques. On the other hand, passive portfolio management avoids any forecasting techniques and rather relies on diversification to achieve a desired performance. There are 2 types of passive management strategies: "buy and hold" or "indexing". In the first one, assets are selected on the basis of some fundamental criteria and there is no active selling or buying of these stocks afterwards (see the sections on Dedication in Chapter 1 and Portfolio Optimization in Chapter 2). In the second approach, absolutely no attempt is made to identify mispriced securities. The goal is to choose a portfolio that mirrors the movements of a broad market population or a market index. Such a portfolio is called an index fund. Given a target population of n stocks, one selects q stocks (and their weights in the index fund), to represent the target population as closely as possible.

In the last twenty years, an increasing number of investors, both large and small, have established index funds. Simply defined, an index fund is a portfolio designed to track the movement of the market as a whole or some selected broad market segment. The rising popularity of index funds can be justified both theoretically and empirically.

- **Market Efficiency:** If the market is efficient, no superior risk-adjusted returns can be achieved by stock picking strategies since the prices reflect all the information available in the marketplace. Additionally, since the market portfolio provides the best possible return per unit of risk, to the extent that it captures the efficiency of the market via diversification, one may argue that the best theoretical approach to fund management is to invest in an index fund.

- **Empirical Performance:** Considerable empirical literature provides strong evidence that, on average, money managers have consistently underperformed the major indexes. In addition, studies show that, in most cases, top performing funds for a year are no longer amongst the top performers in the following years, leaving room for the intervention of luck as an explanation for good performance.

- **Transaction Cost:** Actively managed funds incur transaction costs, which reduce the overall performance of these funds. In addition, active management implies significant research costs. Finally, fund managers may have costly compensation packages that can be avoided to a large extent with index funds.

Strategies for forming index funds involve choosing a broad market index as a proxy for an entire market, e.g. the Standard and Poor list of 500 stocks (S & P 500). A pure indexing approach consists in purchasing all the issues in the index, with the same exact weights as in the index. In most instances, this approach is impractical (many small positions) and expensive (rebalancing costs may be incurred frequently). An index fund with q stocks, where q is substantially smaller than the size n of the target population seems desirable. We propose a large-scale deterministic model for aggregating a broad market index of stocks into a smaller more manageable index fund. This approach will not necessarily yield mean/variance

efficient portfolios but will produce a portfolio that closely replicates the underlying market population.

3.4.1 A Large-Scale Deterministic Model

We present a model that clusters the assets into groups of similar assets and selects one representative asset from each group to be included in the index fund portfolio. The model is based on the following data, which we will discuss in some detail later:

$$\rho_{ij} = \text{similarity between stock } i \text{ and stock } j$$

(For example, $\rho_{ii} = 1$, $\rho_{ij} \leq 1$ for $i \neq j$ and ρ_{ij} is larger for more similar stocks)

$$\begin{array}{lll} (M) & Z = Maximize & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} \rho_{ij}x_{ij} \\ & \text{subject to} & \displaystyle\sum_{j=1}^{n} y_j = q \\ & & \displaystyle\sum_{j=1}^{n} x_{ij} = 1 \qquad \text{for } i = 1,\ldots,n \\ & & x_{ij} \leq y_j \qquad \text{for } i = 1,\ldots,n;\ j = 1,\ldots,n \\ & & x_{ij}, y_j = 0 \text{ or } 1 \qquad \text{for } i = 1,\ldots,n;\ j = 1,\ldots,n. \end{array}$$

The variables y_j describe which stocks j are in the index fund ($y_j = 1$ if j is selected in the fund, 0 otherwise). For each stock $i = 1,\ldots,n$, the variable x_{ij} indicates which stock j in the index fund is most similar to i ($x_{ij} = 1$ if j is the most similar stock in the index fund, 0 otherwise).

Interpret each of the constraints. Explain why the objective of the model can be interpreted as selecting q stocks out of the population of n stocks so that the total loss of information is minimized.

Once the model has been solved and a set of q stocks has been selected for the index fund, a weight w_j is calculated for each j in the fund:

$$w_j = \sum_{i=1}^{n} V_i x_{ij}$$

where V_i is the market value of stock i. So w_j is the total market value of the stocks "represented" by stock j in the fund. The fraction of the index fund to be invested in stock j is proportional to the stock's weight w_j, i.e.

$$\frac{w_j}{\sum_{f \in F} w_f}$$

Note that, instead of the objective function used in (M), one could have used an objective function that takes the weights w_j directly into account, such as $\sum_{i=1}^{n}\sum_{j=1}^{n} V_i \rho_{ij} x_{ij}$. The q

stocks in the index fund found by this variation of Model (M) would still need to be weighted as explained in the previous paragraph.

Data Requirements

We need a coefficient ρ_{ij} which measures the similarity between stocks i and j. There are several ways of constructing meaningful coefficients ρ_{ij}. One approach is to consider the time series of stock prices over a calibration period T and to compute the correlation between each pair of assets.

Testing the Model

Stocks comprising the S&P 500 were chosen as the target population to test the model. A calibration period of sixty months was used. Then a portfolio of 25 stocks was constructed using model (M) and held for periods ranging from three months to three years. The following table gives the ratio of the population's market value (normalized) to the index fund's market value. A perfect index fund would have a ratio equal unity.

Length	Ratio
1 QTR	1.006
2 QTR	.99
1 YR	.985
3 YR	.982

Table 3.3: Performance of a 25 stock index fund

Solution Strategy

Branch-and-bound is a natural candidate for solving model (M). Note however that the formulation is very large. Indeed, for the S&P 500, there are 250,000 variables x_{ij} and 250,000 contraints $x_{ij} \leq y_j$. So the linear programming relaxation needed to get upper bounds in the branch-and-bound algorithm is a very large linear program to solve. It turns out, however, that one does not need to solve this large linear program to obtain good upper bounds. Cornuéjols, Fisher and Nemhauser (*Management Science 23* (1977) pp 789-810) proposed to solve model (M) using the following Lagrangian relaxation, which is defined for any vector $u = (u_1, \ldots, u_n)$:

$$
\begin{array}{lrcll}
L(u) = \text{ Maximize } & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n}\rho_{ij}x_{ij} & + & \displaystyle\sum_{i=1}^{n}u_i(1-\sum_{j=1}^{n}x_{ij}) & \\
\text{subject to} & \displaystyle\sum_{j=1}^{n}y_j & = & q & \\
& x_{ij} & \leq & y_j & \text{for } i=1,\ldots,n \\
& & & & j=1,\ldots,n \\
& x_{ij}, y_j & = & 0 \text{ or } 1 & \text{for } i=1,\ldots,n \\
& & & & j=1,\ldots,n.
\end{array}
$$

Property 1: $L(u) \geq Z$, where Z is the maximum for model (M).

Explain why.

The objective function $L(u)$ may be equivalently stated as

$$L(u) = max \sum_{i=1}^{n}\sum_{j=1}^{n}(\rho_{ij} - u_i)x_{ij} + \sum_{i=1}^{n} u_i.$$

Let

$$(\rho_{ij} - u_i)^+ = \begin{cases} (\rho_{ij} - u_i) & \text{if } \rho_{ij} - u_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

and

$$C_j = \sum_{i=1}^{n}(\rho_{ij} - u_i)^+.$$

Then

Property 2:

$$\begin{aligned} L(u) = & \; max \sum_{j=1}^{n} C_j y_j + \sum_{i=1}^{n} u_i \\ & \text{subject to } \sum_{j=1}^{n} y_j = q \\ & \qquad y_j = 0 \text{ or } 1 \text{ for } j = 1, \ldots, n. \end{aligned}$$

Explain why.

Property 3: In an optimal solution of the Lagrangian relaxation, y_j is equal to 1 for the q largest values of C_j, and the remaining y_j are equal to 0.

If $\rho_{ij} - u_i > 0$, then $x_{ij} = y_j$ and otherwise $x_{ij} = 0$.

Explain why.

Interestingly, the set of q stocks corresponding to the q largest values of C_j can also be used as a heuristic solution for model (M). Indeed, construct an index fund containing these q stocks and assign each stock $i = 1, \ldots, n$ to the most similar stock in this fund. This solution is feasible to model (M), although not necessarily optimal. This heuristic solution provides a lower bound on the optimum value Z of model (M). As previously shown, $L(u)$ provides an upper bound on Z. So for any vector u, we can compute quickly both a lower bound and an upper bound on the optimum value of (M). To improve upon the upper bound $L(u)$, we solve the nonlinear problem

$$\min \; L(u).$$

Since $L(u)$ is nondifferentiable and convex, we employ a subgradient optimization algorithm. At each iteration, a revised set of Lagrange multipliers u and an accompanying lower and upper bound to model (M) are computed. The algorithm terminates when these two bounds match or when a maximum number of iterations is reached. See Chapter 2 for a description of the subgradient method.

3.4.2 A Linear Programming Model

In this section, we consider a different approach to constructing an index fund. It can be particularly useful as one tries to rebalance the portfolio at minimum cost. This approach assumes that we have identified important characteristics of the market index to be tracked. Such characteristics might be the fraction f_i of the index in each sector i, the fraction of companies with market cap in various ranges, the fraction of companies that pay no dividends, the fraction in each region etc. Let us assume that there are m such characteristics that we would like our index fund to track as well as possible. Let $a_{ij} = 1$ if company j has characteristic i and 0 if it does not.

Let x_j denote the optimum weight of asset j in the portfolio. Assume that initially, the portfolio has weights x_j^0. The problem of rebalancing the portfolio at minimum cost is the following:

$$\min \sum_{j=1}^{n} (y_j + z_j)$$

subject to

$$\begin{aligned}
&\sum_{i=1}^{n} a_{ij} x_j = f_i \qquad \text{for } i = 1, \ldots, m \\
&\sum_{j=1}^{n} x_j = 1 \\
&x_j - x_j^0 \leq y_j \qquad \text{for } j = 1, \ldots, n \\
&x_j^0 - x_j \leq z_j \qquad \text{for } j = 1, \ldots, n \\
&y_j \geq 0 \qquad \text{for } j = 1, \ldots, n \\
&z_j \geq 0 \qquad \text{for } j = 1, \ldots, n \\
&x_j \geq 0 \qquad \text{for } j = 1, \ldots, n.
\end{aligned}$$

where y_j denotes the fraction of asset j bought and z_j the fraction sold.

Chapter 4

Dynamic Programming and Structuring CMO's

4.1 Introduction

Let's begin with a simple capital budgeting problem. A corporation has \$5 million to allocate to its three plants for possible expansion. Each plant has submitted a number of proposals on how it intends to spend the money. Each proposal gives the cost of the expansion (c) and the total revenue expected (r). The following table gives the proposals generated:

	Plant 1.		Plant 2.		Plant3	
Proposal	c_1	r_1	c_2	r_2	c_3	r_3
1	0	0	0	0	0	0
2	1	5	2	8	1	4
3	2	6	3	9	—	—
4	—	—	4	12	—	—

Table 4.1: Investment Possibilities

Each plant will only be permitted to enact one of its proposals. The goal is to maximize the firms revenues resulting from the allocation of the \$5 million.

A straightforward way to solve this is to try all possibilities and choose the best. In this case, there are only $3 \times 4 \times 2 = 24$ ways of allocating the money. Many of these are infeasible (for instance, proposals 3, 4, and 1 for the three plants costs \$6 million). Other proposals are feasible, but very poor (like proposals 1, 1, and 2, which is feasible but returns only \$3 million).

Here are some disadvantages of total enumeration:

1. For larger problems the enumeration of all possible solutions may not be computationally feasible.
2. Infeasible combinations cannot be detected *a priori*, leading to inefficiency.
3. Information about previously investigated combinations is not used to eliminate inferior, or infeasible, combinations.

Note also that this problem cannot be formulated as a linear program, for the revenues returned are not linear functions.

One method of calculating the solution is as follows:

Let's break the problem into three **stages**: each stage represents the money allocated to a single plant. So stage 1 represents the money allocated to plant 1, stage 2 the money to plant 2, and stage 3 the money to plant 3. We will artificially place an ordering on the stages, saying that we will first allocate to plant 1, then plant 2, then plant 3.

Each stage is divided into **states**. A state encompasses the information required to go from one stage to the next. In this case the states for stages 1, 2, and 3 are

x_1 = the amount of money spent on plant 1,

x_2 = the amount of money spent on plants 1 and 2, and

x_3 = the amount of money spent on plants 1, 2, and 3.

Associated with each state is a revenue. Note that to make a decision at stage 3, it is only necessary to know how much was spent on plants 1 and 2, not how it was spent. Also notice that we will want x_3 to be 5.

Let's try to figure out the revenues associated with each state. The only easy possibility is in stage 1, the states x_1. Table 4.2 gives the revenue associated with x_1.

If the available capital x_1 is	Then the optimal proposal is	And the revenue for stage 1 is
0	1	0
1	2	5
2	3	6
3	3	6
4	3	6
5	3	6

Table 4.2: Stage 1 computations.

We are now ready to tackle the computations for stage 2. In this case, we want to find the best solution for both plants 1 and 2. If we want to calculate the best revenue for a given x_2, we simply go through all the plant 2 proposals, allocate the given amount of funds to plant 2, and use the above table to see how plant 1 will spend the remainder.

For instance, suppose we want to determine the best allocation if $x_2 = 4$. In stage 2 we can do one of the following proposals:

Proposal 1 gives revenue of 0, leaves 4 for stage 1, which returns 6. Total: 6.

Proposal 2 gives revenue of 8, leaves 2 for stage 1, which returns 6. Total: 14.

Proposal 3 gives revenue of 9, leaves 1 for stage 1, which returns 5. Total: 14.

Proposal 4 gives revenue of 12, leaves 0 for stage 1, which returns 0. Total: 12.

The best thing to do with four units is proposal 1 for plant 2 and proposal 2 for plant 1, returning 14, or proposal 2 for plant 2 and proposal 1 for plant 1, also returning 14. In either case, the revenue for being in state $x_2 = 4$ is 14. The rest of table 4.3 can be filled out similarly:

We can now go on to stage 3. The only value we are interested in is $x_3 = 5$. Once again, we go through all the proposals for this stage, determine the amount of money remaining and

If the available capital x_2 is	Then the optimal proposal is	And the revenue for stages 1 and 2 is
0	1	0
1	1	5
2	2	8
3	2	13
4	2 or 3	14
5	4	17

Table 4.3: Stage 2 computations.

use table 4.3 to decide the value for the previous stages. So here we can do the following at plant 3:

Proposal 1 gives revenue 0, leaves 5. Previous stages give 17. Total: 17.

Proposal 2 gives revenue 4, leaves 4. Previous stages give 14. Total: 18.

Therefore, the optimal solution is to implement proposal 2 at plant 3, proposal 2 or 3 at plant 2, and proposal 3 or 2 (respectively) at plant 1. This gives a revenue of 18.

If you study this procedure, you will find that the calculations are done *recursively*. Stage 2 calculations is based on stage 1. Stage 3 only on stage 2. Indeed, given you are at a state, all future decisions are made independent of how you got to the state. This is the **principle of optimality** and all of dynamic programming rests on this assumption.

We can sum up these calculations in the following formulas:

Denote by $r(k_j)$ the revenue for proposal k_j at stage j, and by $c(k_j)$ the corresponding cost,

and let $f_j(x_j)$ be the revenue of state x_j in stage j.

Then we have the following calculations

$$f_1(x_1) = \max_{k_1 : c(k_1) \leq x_1} \{r(k_1)\}$$

and

$$f_j(x_j) = \max_{k_j : c(k_j) \leq x_j} \{r(k_j) + f_{j-1}(x_j - c(k_j))\} \text{ for } j = 2, 3.$$

All we were doing with the above calculations was determining these functions.

The computations were carried out in a forward procedure. It was also possible to calculate things from the "last" stage back to the first stage. We could define

y_1 = amount allocated to stages 1, 2, and 3,

y_2 = amount allocated to stages 2 and 3, and

y_3 = amount allocated to stage 3.

This defines a **backward recursion**.

Corresponding formulas are:

Let $f_3(y_3)$ be the optimal revenue for stage 3, given y_3,

$f_2(y_2)$ be the optimal revenue for stages 2 and 3, given y_2,

and $f_1(y_1)$ be the optimal revenue for stages 1, 2, and 3, given y_1.

The recursion formulas are:

$$f_3(y_3) = \max_{k_3 : c(k_3) \le y_3} \{r(k_3)\}$$

$$f_j(y_j) = \max_{k_j : c(k_j) \le y_j} \{r(k_j) + f_{j+1}(y_j - c(k_j))\}$$

If you carry out the calculations, you will come up with the same answer.

You may wonder why I have introduced backward recursion, particularly since the forward recursion seems more natural. In this particular case, the ordering of the stages made no difference. In other cases, though, there may be computational advantages of choosing one over another. In general, the backward recursion has been found to be more effective in most applications.

4.2 Characteristics of Dynamic Programming

There are a number of characteristics that are common to all dynamic programming problems. These are

1. The problem can be divided into *stages* with a *decision* required at each stage.

   In the capital budgeting problem the stages were the allocations to a single plant. The decision was how much to spend.

2. Each stage has a number of *states* associated with it.

   The states for the capital budgeting problem corresponded to the amount spent at that point in time.

3. The decision at one stage transforms one state into a state in the next stage.

   The decision of how much to spend gave a total amount spent for the next stage.

4. Given the current state, the optimal decision for each of the remaining states does not depend on the previous states or decisions.

   In the budgeting problem, it is not necessary to know how the money was spent in previous stages, only how much was spent.

5. There exists a recursive relationship that identifies the optimal decision for stage j, given that stage $j + 1$ has already been solved.

6. The final stage must be solvable by itself.

   These last two are tied up in the recursive relationships given above.

The big skill in dynamic programming, and the art involved, is to take a problem and determine stages and states so that all of the above hold. If you can, then the recursive relationship makes finding the values relatively easy. Because of the difficulty in identifying stages and states, we will do a fair number of examples.

4.3 The Knapsack Problem.

The knapsack problem is a particular type of integer program with just one constraint. Each item that can go into the knapsack has a size and a benefit. The knapsack has a certain capacity. What should go into the knapsack so as to maximize the total benefit? As an example, suppose we have three items as follows:

Item (j)	Weight (w_j)	Benefit(b_j)
1	2	65
2	3	80
3	1	30

Table 4.4: Knapsack Items

Suppose the capacity of the knapsack is 5.

4.3.1 A Dynamic Programming Formulation

The stages represent the items: we have three stages $j = 1, 2, 3$. The state y_j at stage j represents the total weight of items j and all following items in the knapsack. The decision at stage j is how many items j to place in the knapsack. Call this value k_j.

This leads to the following recursive formulas: Let $f_j(y_j)$ be the value of using y_j units of capacity for items j and following. Let $\lfloor a \rfloor$ represent the largest integer less than or equal to a.

$f_3(y_j) = 30y_j$

$f_j(y_j) = \max_{k_j \leq \lfloor y_j/w_j \rfloor}\{b_j k_j + f_{j+1}(y_j - w_j k_j)\}$.

4.3.2 An Alternative Dynamic Programming Formulation

There is another formulation for the knapsack problem. This illustrates how arbitrary our definitions of stages, states, and decisions are. It also points out that there is some flexibility on the rules for dynamic programming. Our definitions required a decision at a stage to take us to the next stage (which we would already have calculated through backwards recursion). In fact, it could take us to any stage we have already calculated. This gives us a bit more flexibility in our calculations.

The recursion I am about to present is a forward recursion. For a knapsack problem, let the stages be indexed by w, the weight filled. The decision is to determine the last item added to bring the weight to w. There is just one state per stage. Let $g(w)$ be the maximum benefit that can be gained from a w pound knapsack. Continuing to use b_j and w_j as the weight and benefit, respectively, for item j, the following relates $g(w)$ to previously calculated g values:

$$g(w) = \max_j\{b_j + g(w - w_j)\}$$

Intuitively, to fill a w pound knapsack, we must end off by adding some item. If we add item j, we end up with a knapsack of size $w - w_j$ to fill. To illustrate on the above example:

$g(0) = 0$

$g(1) = 30$ add item 3.

$g(2) = \max\{65 + g(0) = 65, 30 + g(1) = 60\} = 65$ add item 1.
$g(3) = \max\{65 + g(1) = 95, 80 + g(0) = 80, 30 + g(2) = 95\} = 95$ add item 1 or 3.
$g(4) = \max\{65 + g(2) = 130, 80 + g(1) = 110, 30 + g(3) = 125\} = 130$ add item 1.
$g(5) = \max\{65 + g(3) = 160, 80 + g(2) = 145, 30 + g(4) = 160\} = 160$ add item 1 or 3.
This gives a maximum of 160, which is gained by adding 2 of item 1 and 1 of item 3.

4.4 A Model for American Options

For a given stock, let S_k denote its price on day k. We can write

$$S_k = S_{k-1} + X_k$$

where X_k is the change in price from day $k-1$ to day k. The *random walk model* for stock prices assumes that the random variables X_k are independent and identically distributed, and are also independent of the known initial price S_0. We will also assume that the distribution F of X_k has a finite mean.

Now consider an American option on this stock. You can buy the stock at a fixed price c on any day between today (day 0) and day N, when the option expires. You do not have to ever exercise the option, but if you do at a time when the stock price is s, then your profit is $s - c$. What strategy maximizes your expected profit?

Let $f_k(s)$ denote the maximum expected profit when the stock price is s and the option has k additional days before expiration. Here the stages are $k = 1, 2, \ldots$ and the state is s. In contrast to our earlier examples, we do not assume that the state space is finite in this model. Then $f_k(s)$ satisfies the following recursion:

$$f_k(s) = max\{s - c, \int f_{k-1}(s + x)dF(x)\}$$

with the boundary condition

$$f_0(s) = max\{s - c, 0\}.$$

For the case that we are considering (American options), there is no closed form formula for $f_k(s)$. However dynamic programming can be used to compute a numerical solution. In the remainder of this section, we use the recursion formula to derive the structure of the optimal policy.

Exercise 13 *Using induction on k, show that $f_k(s) - s$ is a nonincreasing function of s.*

Solution The fact that $f_0(s) - s$ is a nonincreasing function of s follows from the definition of f_0. Assume now $f_{k-1}(s) - s$ is a nonincreasing function of s. Using the recursion equation, we get

$$f_k(s) - s = max\{-c, \int (f_{k-1}(s + x) - (s + x))dF(x) + \mu\}$$

where μ denotes the mean of F. For any x, the function $f_{k-1}(s+x)-(s+x)$ is a nonincreasing function of s, by the induction hypothesis. It follows that $f_k(s)-s$ is a nonincreasing function of s. **End of solution.**

The optimal policy for an American option has the following form:

There are nondecreasing numbers $s_1 \leq s_2 \leq \ldots \leq s_k \leq \ldots$ such that, if the current stock price is s and there are k days until expiration, then one should exercise the option if and only if $s \geq s_k$.

Let us prove this result. It follows from the recursion equation that if $f_k(s) \leq s - c$, then it is optimal to exercise the option when the stock price is s and there remain k days until expiration. Indeed this yields $f_k(s) = s - c$, which is the maximum possible under the above assumption. Define

$$s_k = min\{s : \ f_k(s) = s - c\}.$$

If no s satisfies $f_k(s) = s - c$, then s_k is defined as $+\infty$. From the exercise above, it follows that

$$f_k(s) - s \leq f_k(s_k) - s_k = -c$$

for any $s \geq s_k$. Therefore it is optimal to exercise the option with k days to expiration whenever $s \geq s_k$. Since $f_k(s)$ is nondecreasing in k, it immediatly follows that s_k is also nondecreasing.

A consequence of the above result is that, when $\mu > 0$, it is always optimal to wait until the maturity date to exercise the option (Explain why). The optimal policy described above becomes nontrivial when $\mu < 0$ however.

4.5 Structuring Collateralized Mortgage Obligations

Mortgages represent the largest single sector of the US debt market, surpassing even the federal government. In 2000, there were over $5 trillion in outstanding mortgages. Because of the enormous volume of mortgages and the importance of housing in the US economy, numerous mechanisms have developed to facilitate the provision of credit to this sector. The predominant method by which this has been accomplished since 1970 is securitization, the bundling of individual mortgage loans into capital market instruments. In 2000, $2.3 trillion of mortgage-backed securities were outstanding, an amount comparable to the $2.1 trillion corporate bond market and $3.4 trillion market in federal government securities.

A mortgage-backed security is a bond backed by a pool of mortgage loans. Principal and interest payments received from the underlying loans are passed through to the bond-holders. These securities contain at least one type of embedded option due to the right of the home buyer to prepay the mortgage loan before maturity. Mortgage-backed securities were first packaged using the pass-through structure. The pass-through's essential characteristic is that investors receive a pro rata share of the cash flows that are generated by the pool of mortgages – interest, scheduled amortization and principal prepayments. Exercise of mortgage prepayment options has pro rata effects on all investors. The pass-through allows banks that initiate mortgages to take their fees up front, and sell the mortgages to investors. One troublesome feature of the pass-through for investors is that the timing and level of the cash flows are uncertain. Depending on the interest rate environment, mortgage holders may pre-pay substantial portions of their mortgage in order to re-finance at lower interest rates.

A collateralized mortgage obligation (CMO) is a more sophisticated mortgage-backed security. The CMO rearranges the cash flows to make them more predictable. This feature

makes CMO's more desirable to investors. The basic idea behind a CMO is to restructure the cashflows from an underlying mortgage collateral (pool of mortgage loans) into a set of bonds with different maturities. These two or more series of bonds (called "tranches") receive sequential, rather than pro rata, principal pay down. Interest payments are made on all tranches (except possibly the last tranche, called Z tranche or "accrual" tranche). A two tranche CMO is a simple example. Assume that there is $100 in mortgages backing two $50 tranches, say tranche A and tranche B. Initially, both tranches receive interest, but principal payments are used to pay down only the A tranche. For example, if $1 in mortgage scheduled amortization and prepayments is collected the first month, the balance of the A tranche is reduced (paid down) by $1. No principal is paid on the B tranche until the A tranche is fully retired. Then the remaining $50 in mortgage principal pays down the $50 B tranche. In effect, the A or "fast-pay" tranche has been assigned all of the early mortgage principal payments (amortization and prepayments) and reaches its maturity sooner than would an ordinary pass-through security. The B or "slow-pay" tranche has only the later principal payments and it begins paying down much later than an ordinary pass-through security.

By repackaging the collateral cashflow in this manner, the life and risk characteristics of the collateral are restructured. The fast-pay tranches are guaranteed to be retired first, implying that their lives will be less uncertain, although not completely fixed. Even the slow-pay tranches will have less cashflow uncertainty than the underlying collateral. Therefore the CMO allows the issuer to target different investor groups more directly than when issuing pass-through securities. The low maturity (fast-pay) tranches may be appealing to investors with short horizons while the long maturity bonds (slow-pay) may be attractive to pension funds and life insurance companies. Each group can find a bond which is better customized to their particular needs.

A by-product of improving the predictability of the cash flows is being able to structure tranches of different credit quality from the same mortgage pool. With the payments of a very large pool of mortgages dedicated to the "fast-pay" tranche, it can be structured to receive a AAA credit rating even if there is a significant default risk on part of the mortgage pool. This high credit rating lowers the interest rate that must be paid on this slice of the CMO. While the credit rating for the early tranches can be very high, the credit quality for later tranches will necessarily be lower because there is less principal left to be repaid and therefore there is increased default risk on slow-pay tranches.

Issuers make money by issuing CMO's because they can pay interest on the tranches that is lower than the interest payments being made by mortgage holders in the pool. The mortgage holders pay 10 or 30-year interest rates on the entire outstanding principal, while some tranches only pay 2, 4, 6 and 8-year interest rates plus an appropriate spread.

The convention in mortgage markets is to price bonds with respect to their weighted average life (WAL), which is much like duration, i.e.

$$WAL = \frac{\sum_{t=1}^{T} t p_t}{\sum_{t=1}^{T} p_t}$$

where p_t is the principal payment in period t $(t = 1, \ldots, T)$.

A bond with a WAL of 3 years will be priced at the 3 year treasury rate plus a spread, while a bond with a WAL of 7 years will be priced at the 7 year treasury rate plus a spread. The WAL of the CMO collateral is typically high, implying a high rate for (normal) upward sloping rate curves. By splitting the collateral into several tranches, some with a low WAL and some with a high WAL, lower rates are obtained on the fast-pay tranches while higher rates result for the slow-pay. Overall, the issuer ends up with a better (lower) average rate on the CMO than on the collateral.

4.5.1 Data

When issuing a CMO, several restrictions apply. First it must be demonstrated that the collateral can service the payments on the issued CMO tranches under several scenarios. These scenarios are well defined and standardized, and cover the two extreme cases of full immediate prepayment of the collateral and no prepayment at all. Second, to price the tranches, the expected WAL of each tranche must lie within certain bounds. This is again caused by the market convention, which allows for a slack in pricing the tranches. Thus a CMO tranche with a WAL between 3 years and 3.44 years is thought of as comparable to a 3 year treasury bond, while one with a WAL between 7 and 7.94 years may be compared to a 7 year treasury bond. The following table contains the payment schedule for a $ 100 Million pool of 10-year mortgages with 10 % interest. It may be useful to remember that, if the outstanding principal is Q, interest is r and amortization occurs over k years, the scheduled amortization in the first year is

$$\frac{Qr}{(1+r)^k - 1}$$

Here $Q = 100$ $r = 0.10$ and $k = 10$, thus the scheduled amortization in the first year is 6.27. Adding the 10 % interest payment on Q, the total payments (interest + scheduled amortization) are $ 16.27 M per year.

Period (t)	Interest (I_t)	Scheduled Amortization (P_t)	Outstanding Principal (Q_t)
1	10.00	6.27	93.73
2	9.37	6.90	86.83
3	8.68	7.59	79.24
4	7.92	8.35	70.89
5	7.09	9.19	61.70
6	6.17	10.11	51.59
7	5.16	11.12	40.47
8	4.05	12.22	28.25
9	2.83	13.45	14.80
10	1.48	14.80	0
Total		100.00	

The above table assumes no prepayment. Next we want to analyze the following scenario: a conditional prepayment model reflecting the 100 % PSA (Public Securities Association) industry-standard benchmark. The rate of mortgage prepayments is 1 % in the first year. Of those mortgages still outstanding, 3 % prepay in the second year. Of those outstanding in the third year, 5 % prepay. Of those outstanding in year t, 6 % prepay in each year $t \geq 4$. For example, in period 1, in addition to the \$10 interest payment and the \$6.27 amortization payment, there is a 1 % prepayment on the 93.73 principal remaining after amortization. That is, there is a \$0.9373 prepayment collected during period 1. Thus the actual principal pay down is $P_1 = 6.27 + 0.9373 = 7.2073$. The outstanding principal after prepayments is $Q_1 = 100 - 7.2073 = 92.7927$. In period 2, the interest paid is $I_2 = 9.279$ and the principal pay down is $P_2 = \frac{Q_1 \times 0.10}{(1.10)^9 - 1} = 9.412$, etc. Construct the table containing I_t, P_t and Q_t to reflect the above scenario.

Loss multiple and required buffer

In order to achieve a high quality rating, tranches should be able to sustain higher than expected default rates without compromising payments to the tranche holders. For this reason, credit ratings are assigned based on how much money is "behind" the current tranche. That is, how much outstanding principal is left after the current tranche is retired, as a percentage of the total amount of principal. This is called the "buffer". Early tranches receive higher credit ratings since they have greater buffers, which means that the CMO would have to experience very large default rates before their payments would be compromised. A tranche with AAA rating must have a buffer equal to six times the expected default rate. This is referred to as the "loss multiple". The loss multiples are as follows:

Credit Rating	AAA	AA	A	BBB	BB	B	CCC
Loss Multiple	6	5	4	3	2	1.5	0

The required buffer is computed by the following formula:

Required Buffer = WAL * Expected Default Rate * Loss Multiple

Based on a 2 % expected default rate assumption, the required buffer to get a AAA rating for a tranche with a WAL of 2 years is $2 \times 0.02 \times 6 = 24$ %. Construct the table containing the required buffer as a function of rating and WAL, assuming a 2 % expected default rate.

Coupon Yields and Spreads

Each tranche is priced based on a credit spread to the current treasury rate for a risk-free bond of that approximate duration. These rates appear in the next table, based on the yields on U.S. Treasuries as of 2/21/2001. You might want to get more current figures. Spreads on corporate bonds with similar credit ratings would provide reasonable figures.

Period (t)	Risk-Free Spot	Credit Spread in Basis Points						
		AAA	AA	A	BBB	BB	B	CCC
1	4.74 %	85	100	115	130	165	220	345
2	4.70 %	90	105	125	140	190	275	425
3	4.77 %	95	110	135	150	210	335	500
4	4.83 %	105	120	145	160	230	380	550
5	4.90 %	115	135	155	175	250	425	625
6	4.94 %	125	145	165	185	265	470	700
7	4.98 %	130	150	175	195	285	515	775
8	5.03 %	135	160	185	210	310	560	850
9	5.07 %	140	165	195	220	330	605	925
10	5.11 %	140	170	200	230	350	650	1000

4.5.2 Characteristics of a tranche that starts amortizing in year j and ends in year t

We are going to consider every possible tranche: since there are 10 possible maturities t and 10 possible starting dates j with $j \leq t$, there are 55 possible tranches. Specifically, tranche (j,t) starts amortizing at the beginning of year j and ends at the end of year t. From the structure of principal payments P_t that you computed earlier, construct the table containing WAL_{jt} for each possible combination (j,t).

For each of the 55 possible tranches (j,t), compute the buffer $\frac{\sum_{k=t+1}^{10} P_k}{\sum_{k=1}^{10} P_k}$. Then calculate the Loss Multiple from the formula: Required Buffer = WAL * Expected Default Rate * Loss Multiple. Finally construct a table containing the credit rating for each of the 55 tranches.

For each of the 55 tranches, construct a table containing the appropriate coupon rate c_{jt}. As described earlier, these rates depend on the WAL and credit rating just computed.

Define Z_{jt} to be the present value of the payments on a tranche (j,t). Armed with the proper coupon rate c_{jt} and a full curve of spot rates r_t, Z_{jt} is computed as follows. In each year k, the payment C_k for tranche (j,t) is equal to the coupon rate c_{jt} times the remaining principal, plus the principal payment made to tranche (j,t) if it is amortizing in year k. The present value of C_k is simply equal to $\frac{C_k}{(1+r_t)^t}$. Now Z_{jt} is obtained by summing the present values of all the payments going to tranche (j,t).

4.5.3 A Dynamic Programming Approach

Based on the above data, we would like to structure a CMO with four sequential tranches A, B, C, Z. The objective is to maximize the profits from the issuance by choosing the size of each tranche.

In this section, we present a dynamic programming recursion for solving the problem.

Let $t = 1, \ldots, 10$ index the years. The states of the dynamic program will be the years t and the stages will be the number k of tranches up to year t.

Now that we have the matrix Z_{jt}, we are ready to describe the dynamic programming

recursion. Let

$f_k(t)$ = Minimum present value of total payments to bondholders in years 1 through t

when the CMO has k tranches up to year t.

Obviously, $f_1(t)$ is simply Z_{1t}. For $k \geq 2$, the value $f_k(t)$ is computed recursively by the formula:

$$f_k(t) = \min_{j=k-1,\ldots,t-1} (f_{k-1}(j) + Z_{j+1,t}).$$

For example, for $k = 2$ and $t = 4$, we compute $f_1(j) + Z_{j+1,4}$ for each $j = 1, 2, 3$ and we take the minimum. The power of dynamic programming becomes clear as k increases. For example, when $k = 4$, there is no need to compute the minimum of thousands of possible combinations of 4 tranches. Instead, we use the optimal structure $f_3(j)$ already computed in the previous stage. So the only enumeration is over the size of the last tranche.

$f_4(10)$ is the least cost value that we are looking for. However, we do not yet have the actual solution that produces this least cost. To find it, we need to backtrack from the last stage and identify how the minimum was acheived at each stage.

4.5.4 An Integer Programming Approach

We optimize n tranches over T periods. To formulate the problem of structuring CMO's as an integer program, we may use the following $2nT$ variables.

p_{it} = the principal payment made to tranche i in period t.

$$z_{it} = \begin{cases} 1 \text{ if tranche } i \text{ is amortized in period } t \\ 0 \text{ otherwise.} \end{cases}$$

For example, if $n = 4$ and $T = 10$, the z_{it} variables might take the following values, indicating that tranche 1 is amortized over years 1 and 2, tranche 2 is amortized over years 3, 4 and 5, etc:

$t =$	1	2	3	4	5	6	7	8	9	10
tranche i=1	1	1	0	0	0	0	0	0	0	0
2	0	0	1	1	1	0	0	0	0	0
3	0	0	0	0	0	1	1	0	0	0
4	0	0	0	0	0	0	0	1	1	1

Next we describe the constraints. We must have

$$\sum_{i=1}^{n} p_{it} = P_t \text{ for } t = 1, \ldots, T.$$

In the above constraints P_t are given data.

We must make sure that the tranches are sequential. To guarantee this, we will assign each P_t to only one tranche, the one that is currently amortized.

$$p_{it} \leq P_t z_{it} \text{ for all } i, t$$

$$\sum_i z_{it} = 1 \text{ for all } t.$$

The second constraint insures that only a single tranche is amortized at a time.

We must also enforce that, once a tranche stops being amortized, it cannot resume amortizing later. This is done by introducing new variables y_{it} that are the sum of z_{jt} variable over $j = 1, \ldots, i$. Namely, sequencing is enforced by

$$\begin{array}{rcll} y_{it} & = & z_{1t} + \ldots + z_{it} & i = 1, \ldots n \\ y_{it} & \geq & y_{it+1} & i = 1, \ldots n. \end{array}$$

In combination, these constraints imply that once a tranche stops being amortized, it cannot resume amortizing later, and that tranches with lower indices i are amortized before tranches with higher indices.

Now let us specify constraints on WAL. Let L_i be a lower bound on the WAL for tranche i and U_i be an upper bound. Then we have

$$L_i \sum_{t=1}^{T} p_{it} \leq \sum_{t=1}^{T} t p_{it} \leq U_i \sum_{t=1}^{T} p_{it}.$$

These constraints will be useful later for adjusting the WAL of each tranche to correspond to its coupon payment.

The payments C_{it} to the bondholders for tranche i in period t consist of coupon payments on the outstanding principal and of principal payments:

$$C_{it} = c_i (\sum_{k=t}^{T} p_{ik}) + p_{it}.$$

Note that the WAL of tranche i is not known yet. Therefore although the c_i's are part of the data, they are guesses based on the expected WAL's of the corresponding tranches. These guesses may have to be revised later. Letting R_t be any excess cash flow from the collateral in any period, we have

$$\sum_i C_{it} + R_t = I_t + P_t$$

The objective is to maximize the excess cash flows from the collateral

$$\sum_{t=1}^{T} \frac{R_t}{(1 + r_t)^t}$$

where r_t denotes the t-year treasury bond rate.

Once the model is set up and solved, we are not done yet. We may need to make adjustments to the model input data: if tranche i has buffer B_i and WAL $= k_i$, then its coupon rate c_i should be equal to the k_i-year treasury rate plus the spread obtained as

computed earlier. If this is not the case, then c_i needs to be adjusted accordingly and the program resolved. To speed convergence of this iterative process, one can tighten the ranges $[L_i, U_i]$ in the model. Finally, remember that we must ensure that the CMO can be retired under best and worst case scenarios. These best and worst case constraints can be treated parametrically and need not be part of the optimization model itself.

Chapter 5

Stochastic Programming and Asset/Liability Management

5.1 Introduction

The term *stochastic program* refers to a mathematical program in which some problem data are random. The underlying mathematical program might be a linear program, an integer program or a nonlinear program. An important case is that of *stochastic linear programs*.

A stochastic program *with recourse* arises when some of the decisions (recourse actions) can be taken after the outcomes of some (or all) random events have become known. For example, a *two-stage stochastic linear program with recourse* can be written as follows:

$$
\begin{array}{lll}
(1) \quad \max & c^1x^1 + E[\max\ c^2(\omega)x^2(\omega)] & \\
\text{subject to} & & \\
& A^1x^1 & = b^1 \\
& B^2(\omega)x^1 \quad +A^2(\omega)x^2(\omega) & = b^2(\omega) \\
& x^1 \geq 0, \quad x^2(\omega) \geq 0 &
\end{array}
$$

where the first-stage decisions are represented by vector x^1 and the second-stage decisions by vector $x^2(\omega)$, which depends on the realization ω of a random event. A^1 and b^1 define deterministic constraints on the first-stage decisions x^1, whereas $A^2(\omega)$, $B^2(\omega)$ and $b^2(\omega)$ define stochastic constraints linking the recourse decisions $x^2(\omega)$ to the first-stage decisions. The objective function contains a deterministic term c^1x^1 and the expectation of the second-stage objective $c^2(\omega)x^2(\omega)$ taken over all realizations of the random event ω.

Note that, once the first-stage decisions x^1 have been made and the random event ω has been realized, one can compute the optimal second-stage decisions by solving the following linear program:

$$
\begin{array}{lll}
f(x^1,\omega) = \max & c^2(\omega)x^2(\omega) & \\
\text{subject to} & & \\
& A^2(\omega)x^2(\omega) & = b^2(\omega) - B^2(\omega)x^1 \\
& x^2(\omega) \geq 0. &
\end{array}
$$

Let $f(x^1) = E[f(x^1,\omega)]$ denote the expected value of $f(x^1,\omega)$. Then the two-stage stochastic linear program becomes

$$\begin{array}{rl} \max & c^1x^1 + f(x^1) \\ \text{subject to} & \\ & A^1x^1 = b^1 \\ & x^1 \geq 0. \end{array}$$

So, if the function f is known, the problem reduces to a nonlinear program. When the data $c^2(\omega), A^2(\omega), B^2(\omega)$ and $b^2(\omega)$ are described by finite distributions, one can show that f is piecewise linear and concave. When the data are described by probability densities that are absolutely continuous and have finite second moments, one can show that f is differentiable and concave. In both cases, we have a convex program with linear constraints for which specialized algorithms are available.

Instead of solving a sequence of mathematical programs (an LP and an NLP in the 2-stage stochastic program example described above), another approach consists in discretizing the random events and formulating the 2-stage stochastic program as a single large scale LP. Let $\omega_1, \omega_2, \ldots, \omega_K$ be the possible realizations of the random event and p_k the corresponding probabilities. Then (1) can be rewritten as

$$\begin{array}{rl} (1') \quad \max & c^1x^1 + \sum_{k=1}^{K} p_k c^2(\omega_k)x^2(\omega_k) \\ \text{subject to} & \\ & \begin{array}{rcll} A^1x^1 & & = & b^1 \\ B^2(\omega_k)x^1 & +A^2(\omega_k)x^2(\omega_k) & = & b^2(\omega_k) \quad \text{for } k = 1, \ldots K \\ x^1 \geq 0, & x^2(\omega_k) \geq 0 & & \end{array} \end{array}$$

where the first stage decision variables are x^1 as before but, in the second stage, there are K different vectors of decision variables $x^2(\omega_k)$ for $k = 1, \ldots, K$. Note that all the data are now deterministic. Therefore, the program (1') is a linear program. Linear programs can be solved with current software even they have thousands of constraints and variables.

The 2-stage example above can be generalized to a *multi-stage stochastic program with recourse* formulated as follows:

$$\begin{array}{rl} (2) \quad \max & c^1x^1 + E_{\omega^2}[\max\,(c^2(\omega^2)x^2(\omega^2) + \ldots + E_{\omega^T|\omega^{T-1}}[\max(c^T(\omega^T)x^T(\omega^T))]\ldots)] \\ \text{subject to} & \\ & \begin{array}{rccccc} A^1x^1 & & & & & = b^1 \\ B^2(\omega^2)x^1 & +A^2(\omega^2)x^2(\omega^2) & & & & = b^2(\omega^2) \\ & B^3(\omega^3)x^2(\omega^2) & +A^3(\omega^3)x^3(\omega^3) & & & = b^3(\omega^3) \\ & & & \ldots & & \\ & & B^T(\omega^T)x^{T-1}(\omega^{T-1}) & +A^T(\omega^T)x^T(\omega^T) & = & b^T(\omega^T) \\ x^1 \geq 0, & x^2(\omega^2) \geq 0, & \ldots & x^T(\omega^T) \geq 0 & & \end{array} \end{array}$$

where the first-stage decisions are represented by vector x^1 and the stage t decisions by vector $x^t(\omega^t)$, for $t = 2, \ldots, T$. In the objective, $E_{\omega^t|\omega^{t-1}}$ denotes the conditional expectation of the state ω^t of the data process $\omega = (\omega^2, \ldots, \omega^T)$ at time t given the history of the process up to time $t-1$. The sequence of events is the following:

$$\text{decide } x^1 \quad \text{observe } \omega^2 \quad \text{decide } x^2(\omega^2) \quad \ldots \quad \text{observe } \omega^T \quad \text{decide } x^T(\omega^T).$$

Using the first approach presented for the two-stage case, one can reduce multistage stochastic programs to a nested sequence of nonlinear programs. Computationally, another

approach is receiving increasing attention, in the spirit if the second approach presented above for the two-stage case. It is called the *scenario approach*: a finite number of possible realizations of the future outcomes are considered. One can think of the scenarios as a tree with T levels where, at each stage $t = 1, \ldots, T-1$, the branches correspond to refinements of ω^t to ω^{t+1}. The stochastic program can then be written as a very large linear program as follows. For $t = 2, \ldots, T$, let $p_{k_2,k_3,\ldots,k_t}$ denote the conditional probability of outcome k_t at stage t given outcomes $k_2, \ldots, k_{t-1}$ at the previous stages. Then (2) can be written as

$$
\begin{array}{lll}
(3) \quad \max & c^1x^1 + \sum_{k_2=1}^{K_2}[p_{k_2}(c^2_{k_2}x^2_{k_2} + \ldots + \sum_{k_T=1}^{K_T}[p_{k_2,\ldots,k_T}c^T_{k_2,\ldots,k_T}x^T_{k_2,\ldots,k_T}]\ldots)] & \\
\text{subject to} & & \\
& A^1x^1 & = b^1 \\
& B^2_{k_2}x^1 + A^2_{k_2}x^2_{k_2} & = b_{k_2} \\
& B^3_{k_2k_3}x^2_{k_2} + A^3_{k_2k_3}x^3_{k_2k_3} & = b_{k_2k_3} \\
& \cdots & \\
& B^T_{k_2,\ldots,k_T}x^{T-1}_{k_2,\ldots,k_{T-1}} + A^T_{k_2,\ldots,k_T}x^T_{k_2,\ldots,k_T} & = b^T_{k_2,\ldots,k_T} \\
& x^1 \geq 0, \quad x^2_{k_2} \geq 0, \quad \ldots \quad x^T_{k_2,\ldots,k_T} \geq 0 &
\end{array}
$$

where $B^t_{k_2,\ldots,k_t}$ denotes the realization of the matrix $B^t(\omega)$ when outcome k_2 occurs followed by outcome k_3 etc ... followed by outcome k_t. The matrices $A^t_{k_2,\ldots,k_t}$ and vectors $b^t_{k_2,\ldots,k_t}$ are similarly defined. There is a constraint $B^2_{k_2}x^1 + A^2_{k_2}x^2_{k_2} = b_{k_2}$ for every $k_2 = 1, \ldots, K_2$ and, more generally, for $t = 2, \ldots, T$, there is a constraint $B^t_{k_2,\ldots,k_t}x^{t-1}_{k_2,\ldots,k_{t-1}} + A^t_{k_2,\ldots,k_t}x^t_{k_2,\ldots,k_t} = b^t_{k_2,\ldots,k_t}$ for every $k_2 = 1, \ldots, K_2$, ..., $k_t = 1, \ldots, K_t$. It should be clear from this construction that (3) is a very large scale linear program. For example, for a problem with 10 stages and a binary tree, there are 1024 scenarios and therefore (3) has several thousand constraints and variables. Modern commercial codes can handle such large linear programs, but a moderate increase in the number of stages or in the number of branches at each stage could make (3) too large to solve.

5.2 Asset/Liability Management

How should a financial institution manage its assets? A static mean-variance model fails to incorporate the multiperiod nature of the liabilities faced by the company. Furthermore, it equally penalizes returns above the mean and shortfalls. A multiperiod model that emphasizes the need to meet liabilities in each period seems more appropriate. The dedicated bond portfolio example (Project 1) was in this vein. However, the deterministic nature of the formulation misses a key aspect of the problem, namely that the liabilities and asset returns usually have a random component. Multistage stochastic programming addresses these issues.

Let L_t be the liability of the company in year t for $t = 1, \ldots, T$. The L_t's are random with known distributions. Given these liabilities, which assets (and in which quantities) should the company hold each year to maximize its expected wealth in year T? The assets may be US stocks, foreign stocks, real estate, bonds, etc. Let R_{it} denote the return on asset i in year t. The R_{it}'s are random with known distributions. The decision variables are:

$$x_{it} = \text{ market value of investment in asset } i \text{ in year } t.$$

The decisions x_{it} in year t are made after the random values L_t and R_{it} have been realized, i.e. after the values of L_t and R_{it} are known with certainty. Thus the decision problem is multistage, stochastic, with recourse. The stochastic program can be written as follows.

$$\begin{array}{rl} \max & E[\sum_i x_{iT}] \\ \text{subject to} & \\ \text{asset accumulation:} & \sum_i (1+R_{it})x_{i,t-1} - \sum_i x_{it} = L_t \quad \text{for } t = 1,\ldots,T \\ & x_{it} \geq 0. \end{array}$$

The constraint says that the surplus left after liability L_t is covered will be invested as follows: x_{it} invested in asset i. The objective selected in the above model was to maximize the expected wealth at the end of the planning horizon. In practice, one might have a different objective. In some cases, minimizing Value at Risk (VaR) might be more appropriate. We have seen in Section 2.11 how one can sometimes linearize the VaR objective by introducing new variables. Other priorities may dictate other objective functions.

To address the issue of the most appropriate objective function, one must understand the role of liabilities. We consider the case of a japanese insurance company, the Yasuda Fire and Marine Insurance Co, Ldt. In this case the liabilities are mainly savings-oriented policies issued by the company. Each new policy sold represents a deposit, or inflow of funds. Interest is periodically credited to the policy until maturity, typically three to five years, at which time the principal amount plus credited interest is refunded to the policyholder. The crediting rate is typically adjusted each year in relation to a market index like the prime rate. Therefore, we cannot say with certainty what future liabilities will be. Insurance business regulations stipulate that interest credited to some policies be earned from investment interest income, not capital gains. So, in addition to ensuring that the maturity cash flows are met, the firm must seek to avoid interim shortfalls in interest income earned versus interest credited in any period. In fact, it is the risk of not earning adequate interest income quarter by quarter that the decision makers view as the primary component of risk at Yasuda.

The problem is to determine the optimal allocation of the deposited funds into several asset categories: cash, fixed rate and floating rate loans, bonds, equities, real estate and other assets. Since we can revise the portfolio allocations over time, the decision we make is not just among allocations today but among allocation strategies over time. A realistic dynamic asset/liability model must also account for the payment of taxes. This is made possible by distinguishing between income return and price return.

A stochastic linear program as in (2) is used to model the problem. The linear program has uncertainty in many coefficients. This uncertainty is modeled through a finite number of scenarios. In this fashion, the problem is transformed into a very large scale linear program of the form (3). The random elements include price and income returns for each asset class, as well as the rates at which interest is credited to the policies issued by the company.

We now present a multistage stochastic program that was developed for The Yasuda Fire and Marine Insurance Co., Ldt. Our presentation follows the description of the model as stated in

D.R. Cariño, T. Kent, D.H. Myers, C. Stacy, M. Sylvanus, A.L. Turner, K. Watanabe, W. Ziemba, The Russell-Yasuda Kasai Model: An Asset/Liability Model for a Japanese Insurance Company Using Multistage Stochastic Programming, *Interfaces 24* (1994) 29-49.

Stages are indexed by $t = 0, 1, \ldots, T$.

Decision variables of the stochastic program:

$$\begin{aligned} x_{it} &= \text{market value in asset } i \text{ at time } t \\ w_t &= \text{interest income shortfall at } t \geq 1 \\ v_t &= \text{interest income surplus at } t \geq 1 \end{aligned}$$

Random data appearing in the stochastic linear program: For $t \geq 1$,

$$\begin{aligned} RP_{it} &= \text{price return of asset } i \text{ from } t-1 \text{ to } t \\ RI_{it} &= \text{income return of asset } i \text{ from } t-1 \text{ to } t \\ F_t &= \text{deposit inflow from } t-1 \text{ to } t \\ P_t &= \text{principal payout from } t-1 \text{ to } t \\ I_t &= \text{interest payout from } t-1 \text{ to } t \\ g_t &= \text{rate at which interest is credited to policies from } t-1 \text{ to } t \\ L_t &= \text{liability valuation at } t \end{aligned}$$

Parametrized function appearing in the objective:

$$c_t = \text{piecewise linear convex cost function}$$

The objective of the model is to allocate funds among available assets to maximize expected wealth at the end of the planning horizon T less expected penalized shortfalls accumulated through the planning horizon.

$$\begin{array}{rl} (4) \quad \max & E[\sum_i x_{iT} - \sum_{t=1}^{T} c_t(w_t)] \\ \text{subject to} & \\ \text{asset accumulation:} & \sum_i x_{it} - \sum_i (1 + RP_{it} + RI_{it}) x_{i,t-1} = F_t - P_t - I_t \quad \text{for } t = 1, \ldots, T \\ \text{income shortfall:} & \sum_i RI_{it} x_{i,t-1} + w_t - v_t = g_t L_{t-1} \quad \text{for } t = 1, \ldots, T \\ & x_{it} \geq 0, \quad w_t \geq 0, \quad v_t \geq 0. \end{array}$$

Liability balances and cash flows are computed so as to satisfy the liability accumulation relations.

$$L_t = (1 + g_t) L_{t-1} + F_t - P_t - I_t \quad \text{for } t \geq 1.$$

The stochastic linear program (4) is converted into a large linear program using a finite number of scenarios to deal with the random elements in the data. Creation of scenario inputs is made in stages using a tree. The tree structure can be described by the number of branches at each stage. For example, a 1-8-4-4-2-1 tree has 256 scenarios. Stage $t = 0$ is the initial stage. Stage $t = 1$ may be chosen to be the end of Quarter 1 and has 8 different scenarios in this example. Stage $t = 2$ may be chosen to be the end of Year 1, with each of the previous scenarios giving rise to 4 new scenarios, and so on. For the Yasuda Fire and Marine Insurance Co., Ldt., a problem with 7 asset classes and 6 stages gives rise to a stochastic linear program (4) with 12 constraints (other than nonnegativity) and 54 variables. Using 256 scenarios, this stochastic program is converted into a linear program with several thousand constraints and over 10,000 variables. Solving this model yielded extra income estimated to about US$ 80 million per year for the company.

5.3 Option Pricing in the Presence of Transaction Costs

A European call option on a stock with maturity T and strike price X gives the right to buy the stock at price X at time T. The holder of the option will not exercise this option if the stock has a price S lower than X at time T. Therefore the value of a European call option is $\max(S - X, 0)$. Since S is random, the question of pricing the option correctly is of interest. The Black-Sholes Option Pricing model relates the price of an option to the volatility of the stock return. The assumptions are that the market is efficient and that the returns are lognormal. From the volatility σ of the stock return, one can compute the option price for any strike price X. Conversely, from option prices one can compute the implied volatility σ. For a given stock, options with different strike prices should lead to the same σ (if the assumptions of the Black-Scholes model are correct).

The aim of the model developed in this section is to examine the extent to which market imperfections can explain the deviation of observed option prices from the Black-Scholes Option Pricing model. One way to measure the deviation of the Black-Scholes model from observed option prices is through the "volatility smile": for a given maturity date, the implied volatility of a stock computed by the Black-Sholes model from observed option prices at different strike prices is typically not constant, but instead often exhibits a convex shape as the strike price increases (the "smile"). One explanation for the deviation is that the smile occurs because the Black-Scholes model assumes the ability to rebalance portfolios without costs imposed either by the inability to borrow or due to a bid-ask spread or other trading costs. Here we will look at the effect of transaction costs on option prices.

The derivation of the Black-Scholes formula is through a replicating portfolio containing the stock and a riskless bond. If the market is efficient, we should be able to replicate the option payoff at time T by rebalancing the portfolio between now and time T, as the stock price evolves. Rather than work with a continuous time model, we discretize this process. This discretization is called the binomial approximation to the Black-Scholes Option Pricing model. In this model, we specify a time period Δ between trading oportunities and postulate the behavior of stock and bond prices along successive time periods. The binomial model assumes that in between trading periods, only two possible stock price movements are possible.

a) There are N stages in the tree, indexed $0, 1 \ldots, N$, where stage 0 is the root of the tree and stage N is the last stage. If we divide the maturity date T of an option by N, we get that the length of a stage is $\Delta = T/N$.

b) Label the initial node k_0.

c) For a node $k \neq k_0$, let k^- be the node that is the immediate predecessor of k.

d) Let $S(k)$ be the stock price at node k and let $B(k)$ be the bond price at node k.

e) We assume that the interest rate is fixed at the annualized rate r so that $B(k) = B(k^-)e^{r\Delta}$.

f) Letting σ denote the volatility of the stock return, we use the standard parametrization $u = e^{\sigma\sqrt{\Delta}}$ and $d = 1/u$. So $S(k) = S(k^-)e^{\sigma\sqrt{\Delta}}$ if an uptick occurs from k^- to k and $S(k) = S(k^-)e^{-\sigma\sqrt{\Delta}}$ if a downtick occurs.

g) Let $n(k)$ be the quantity of stocks at node k and let $m(k)$ be the quantity of bonds at k.

5.3.1 The Standard Problem

In the binomial model, we have dynamically complete markets. This means that by trading the stock and the bond dynamically, we can replicate the payoffs (and values) from a call option. The option value is simply the cost of the replicating portfolio, and the replicating portfolio is self-financing after the first stage. This means that after we initially buy the stock and the bond, all subsequent trades do not require any additional money and, at the last stage, we reproduce the payoffs from the call option.

Therefore, we can represent the option pricing problem as the following linear program. Choose quantities $n(k)$ of the stock, quantities $m(k)$ of the bond at each nonterminal node k to

$$\begin{array}{rl} (5)\quad \min & n(k_0)S(k_0) + m(k_0)B(k_0) \\ \text{subject to} & \\ \text{rebalancing constraints:} & n(k^-)S(k) + m(k^-)B(k) \geq n(k)S(k) + m(k)B(k) \\ & \text{for every node } k \neq k_0 \\ \text{replication constraints:} & n(k)S(k) + m(k)B(k) \geq \max(S(k) - X, 0) \\ & \text{for every terminal node } k \end{array}$$

where k^- denotes the predecessor of k.

Note that we do not impose nonnegativity constraints since we will typically have a short position in the stock or bond.

Exercise: Collect data on 4 or 5 call options on a nondividend paying stock for the nearest maturity (but at least one month). Calculate the implied volatility for each option. Solve the standard problem (5) when the number of stages is 7 using the implied volatility of the at-the-money option to construct the tree.

5.3.2 Transaction Costs

To model transaction costs, we consider the simplest case where there are no costs of trading at the initial and terminal nodes, but there is a bid-ask spread on stocks at other nodes. So assume that if you buy a stock at node k, you pay $S(k)(1+\theta)$ while if you sell a stock, you receive $S(k)(1-\theta)$. This means that the rebalancing constraint becomes

$$n(k^-)S(k) + m(k^-)B(k) \geq n(k)S(k) + m(k)B(k) + |n(k) - n(k^-)|\theta S(k).$$

There is an absolute value in this constraint. So it is not a linear constraint. However it can be linearized as follows. Define two nonnegative variables:

$$\begin{array}{rcl} x(k) & = & \text{number of stocks } \textit{bought} \text{ at node } k, \text{ and} \\ y(k) & = & \text{number of stocks } \textit{sold} \text{ at node } k. \end{array}$$

The rebalancing constraint now becomes:

$$n(k^-)S(k) + m(k^-)B(k) \geq n(k)S(k) + m(k)B(k) + (x(k) + y(k))\theta S(k)$$
$$n(k) - n(k^-) = x(k) - y(k)$$
$$x(k) \geq 0, \quad y(k) \geq 0.$$

Note that this constraint leaves the possibility of simultaneously buying and selling stocks at the same node. But obviously this cannot improve the objective function that we minimize in (5), so we do not need to impose a constraint to prevent it.

The modified formulation is:

$$\begin{array}{rll}
(6) \quad \min & n(k_0)S(k_0) + m(k_0)B(k_0) & \\
\text{subject to} & & \\
\text{rebalancing constraints:} & n(k^-)S(k) + m(k^-)B(k) \geq n(k)S(k) + m(k)B(k) & \\
& +(x(k) + y(k))\theta S(k) \quad \text{for every node } k \neq k_0 & \\
& n(k) - n(k^-) = x(k) - y(k) \quad \text{for every node } k \neq k_0 & \\
\text{replication constraints:} & n(k)S(k) + m(k)B(k) \geq \max(S(k) - X, 0) & \\
& \text{for every terminal node } k & \\
\text{nonnegativity:} & x(k) \geq 0, \quad y(k) \geq 0 \quad \text{for every node } k \neq k_0. &
\end{array}$$

Exercise: Repeat the exercise in Section 5.3.1 allowing for transaction costs, with different values of θ, to see if the volatility smile can be explained by transaction costs. Specifically, given a value for σ and for θ, calculate option prices and see how they match up to observed prices.

5.4 Synthetic Options

An important issue in portfolio selection is the potential decline of the portfolio value below some critical limit. How can we control the risk of downside losses? A possible answer is to create a payoff structure similar to a European call option.

While one may be able to construct a diversified portfolio well suited for a corporate investor, there may simply be no option market available when it comes to options on this portfolio. One solution for investors may be to use index options. However exchange-traded options with sufficient liquidity are limited to maturities of about three months. This makes the cost of long-term protection expensive, requiring the purchase of a series of high priced short-term options. For large institutional or corporate investors, a cheaper solution is to synthetize the desired payoff structure using available resources. This is called a "synthetic option strategy".

The model is based on the following data.

$$\begin{array}{rcl}
W_0 & = & \text{investor's initial wealth} \\
T & = & \text{planning horizon} \\
r & = & \text{riskless rate of return for one period} \\
r_t^i & = & \text{rate of return for asset i at time } t \\
\theta_t^i & = & \text{transaction cost for purchases and sales of asset i at time } t.
\end{array}$$

The r_t^i's are random, but we know their distributions.

The variables used in the model are the following.

x_t^i = amount allocated to asset i at time t
A_t^i = amount of asset i bought at time t
D_t^i = amount of asset i sold at time t
α_t = amount allocated to riskless asset at time t.

We formulate a stochastic program that produces the desired payoff at the end of the planning horizon T, much in the flavor of the stochastic programs developed in the previous two sections. Let us first discuss the constraints.

The initial portfolio is

$$\alpha_0 + x_0^1 + \ldots + x_0^n = W_0.$$

The portfolio at time t is

$$x_t^i = x_{t-1}^i e^{r_t^i} + A_t^i - D_t^i \quad \text{for } t = 1, \ldots, T$$

$$\alpha_t = \alpha_{t-1} e^r - \sum_{i=1}^{n} (1 + \theta_t^i) A_t^i + \sum_{i=1}^{n} (1 - \theta_t^i) D_t^i \quad \text{for } t = 1, \ldots, T.$$

Note that, in the equation defining x_t^i above, the term $e^{r_t^i}$ is the random return of asset i at time t.

One can also impose upper bounds on the proportion of any risky asset in the portfolio:

$$0 \leq x_t^i \leq m_t(\alpha_t + \sum_{j=1}^{n} x_t^j),$$

where m_t is chosen by the investor.

The value of the portfolio at the end of the planning horizon is:

$$v = \alpha_{T-1} e^r + \sum_{i=1}^{n} (1 - \theta_T^i) x_{T-1}^i e^{r_T^i},$$

where the summation term is the value of the risky assets at time T.

To construct the desired synthetic option, we split v into the riskless value of the portfolio R and a surplus $z \geq 0$ which depends on random events. Using a scenario approach to the stochastic program, R is the worst-case payoff over all the scenarios. The surplus z is a random variable that depends on the scenario. Thus

$$v = R + z$$

$$z \geq 0.$$

The objective function of the stochastic program is

$$\max E(z) + \mu R$$

where $\mu \geq 1$ is the risk aversion of the investor.

When $\mu = 1$, the objective is to maximize expected return.

When μ is very large, the objective is to maximize "riskless profit" as we defined it in Chapter 1 (Section 1.4.1).

As an example, consider an investor with initial wealth $W_0 = 1$ who wants to construct a portfolio comprising one risky asset and one riskless asset using the "synthetic option" model described above. We write the model for a two-period planning horizon, i.e. $T = 2$. The rate of return on the riskless asset is r per period. For the risky asset, the rate of return is r_1^+ with probability .5 and r_1^- with the same probability at time $t = 1$. Similarly, the rate of return of the risky asset is r_2^+ with probability .5 and r_2^- with the same probability at time $t = 2$. The transaction cost for purchases and sales of the risky asset is θ.

There are 4 scenarios in this example, each occuring with probability .25, which we can represent by a binary tree. The initial node will be denoted by 0, the up node from it by 1 and the down node by 2. Similarly the up node from node 1 will be denoted by 3, the down node by 4, and the successors of 2 by 5 and 6 respectively. Let x_i, α_i denote the amount of risky asset and of riskless asset respectively in the portfolio at node i of this binary tree. R is the riskless value of the portfolio and z_i is the surplus at node i. The linear program is:

$$
\begin{aligned}
\max \quad & .25z_3 + .25z_4 + .25z_5 + .25z_6 + \mu R \\
\text{subject to} & \\
\text{initial portfolio:} \quad & \alpha_0 + x_0 = 1 \\
\text{rebalancing constraints:} \quad & x_1 = x_0 e^{r_1^+} + A_1 - D_1 \\
& \alpha_1 = \alpha_0 e^r - (1+\theta)A_1 + (1-\theta)D_1 \\
& x_2 = x_0 e^{r_1^-} + A_2 - D_2 \\
& \alpha_2 = \alpha_0 e^r - (1+\theta)A_2 + (1-\theta)D_2 \\
\text{payoff:} \quad & z_3 + R = \alpha_1 e^r + (1-\theta)x_1 e^{r_2^+} \\
& z_4 + R = \alpha_1 e^r + (1-\theta)x_1 e^{r_2^-} \\
& z_5 + R = \alpha_2 e^r + (1-\theta)x_2 e^{r_2^+} \\
& z_6 + R = \alpha_2 e^r + (1-\theta)x_2 e^{r_2^-} \\
\text{nonnegativity:} \quad & \alpha_i, x_i, z_i \geq 0.
\end{aligned}
$$

Example: An interesting paper discussing synthetic options is the paper of Y. Zhao and W.T. Ziemba entitled "A stochastic programming model using an endogenously determined worst case risk measure for dynamic asset allocation" published in *Mathematical Programming B 89* (2001) pages 293-309. Zhao and Ziemba apply the synthetic option model to an example with 3 assets (cash, bonds and stocks) and 4 periods (a one-year horizon with quarterly portfolio reviews). The quaterly return on cash is constant at $\rho = 0.0095$. For stocks and bonds, the expected logarithmic rates of returns are $s = 0.04$ and $b = 0.019$ respectively. Transaction costs are 0.5% for stocks and 0.1% for bonds. The scenarios needed in the stochastic program are generated using an auto regression model which is constructed based on historical data (quaterly returns from 1985 to 1998; the Salomon Brothers bond index and S&P 500 index respectively). Specifically, the auto regression model is

$$
\begin{cases}
s_t = 0.037 - 0.193 s_{t-1} + 0.418 b_{t-1} - 0.172 s_{t-2} + 0.517 b_{t-2} + \epsilon_t \\
b_t = 0.007 - 0.140 s_{t-1} + 0.175 b_{t-1} - 0.023 s_{t-2} + 0.122 b_{t-2} + \eta_t
\end{cases}
$$

where the pair (ϵ_t, η_t) characterizes uncertainty. The scenarios are generated by selecting 20 pairs of (ϵ_t, η_t) to estimate the empirical distribution of one period uncertainty. In this way,

a scenario tree with 160,000 ($= 20 \times 20 \times 20 \times 20$) paths describing possible outcomes of asset returns is generated.

The resulting large scale linear program is solved. We discuss the results obtained when this linear program is solved for a risk aversion of $\mu = 2.5$: The value of the terminal portfolio is always at least 4.6% more than the initial portfolio wealth and the distribution of terminal portfolio values is skewed to larger values because of dynamic downside risk control. The expected return is 16.33% and the volatility is 7.2%. It is interesting to compare these values with those obtained from a static Markowitz model: The expected return is 15.4% for the same volatility but no minimum return is guaranteed! In fact, in some scenarios, the value of the Markowitz portfolio is 5% *less* at the end of the one-year horizon than it was at the beginning.

It is also interesting to look at an example of a typical portfolio (one of the 160,000 paths) generated by the synthetic option model (the linear program was set up with an upper bound of 70 % placed on the fraction of stocks or bonds in the portfolio):

	Cash	Stocks	Bonds	Portfolio value
				100
Period1	12%	18%	70%	103
2		41%	59%	107
3		70%	30%	112
4	30%		70%	114

Exercise: Develop a synthetic option model in the spirit of that used by Zhao and Ziemba, adapted to the size limitation of your linear programming solver. Compare with a static model.