Nonlinear Optimization

ADAPTED FROM AN INTRODUCTION TO MANAGEMENT SCIENCE BY ANDERSON ET. AL & QUANTITATIVE ANALYSIS FOR MANAGEMENT BY RENDER ET. AL COPYRIGHT: CENAGE PUBLISHING AND PEARSON EDUCATION

Nonlinear Optimization

1- A Production Application – Par, Inc.

- LP formulation & solution
- Nonlinear formulation & solution

2- CVXOPT library installation and application

2 – Markowitz Portfolio Model

- Markowitz implementation using CVXOPT

Introduction

Many business processes behave in a nonlinear manner.

- The price of a bond is a nonlinear function of interest rates.
- The price of a stock option is a nonlinear function of the price of the underlying stock.
- The marginal cost of production often decreases with the quantity produced.
- The quantity demanded for a product is often a nonlinear function of the price.

A **nonlinear optimization problem** is any optimization problem in which at least one term in the objective function or a constraint is nonlinear.

A Simple Maximization Problem (Par Inc. linear case)

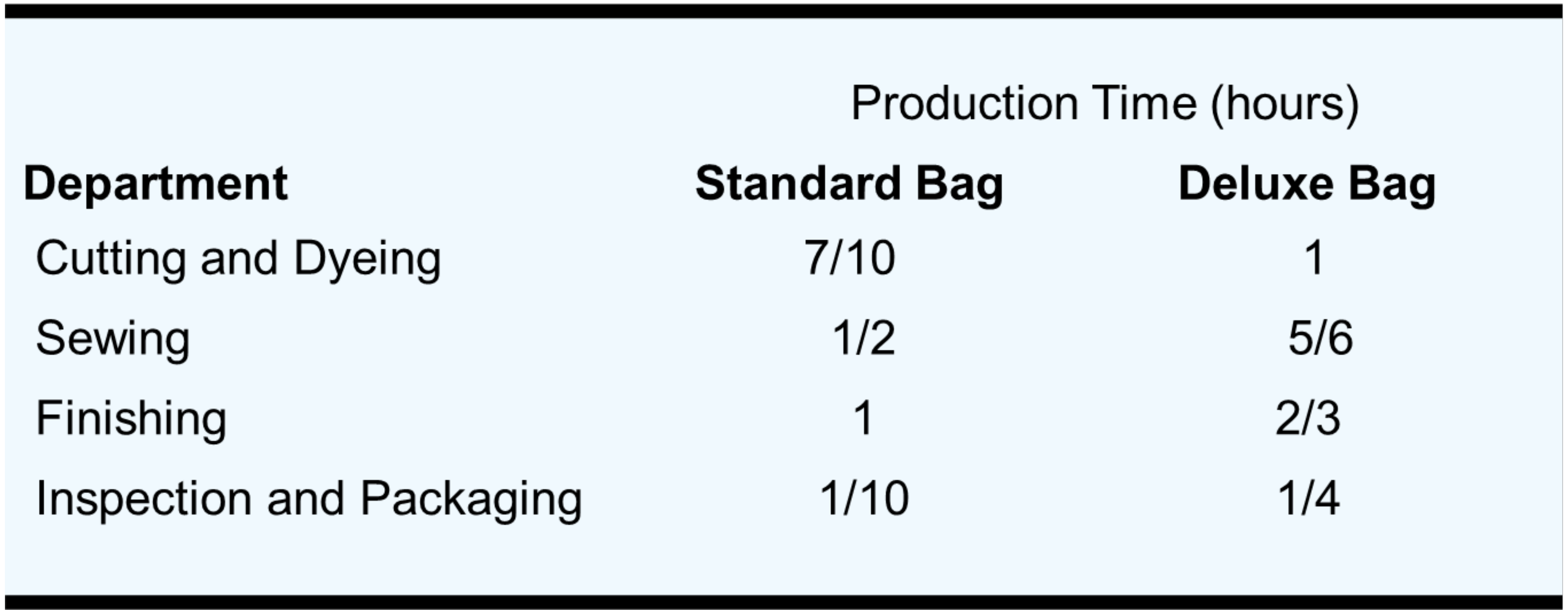
Par, Inc., is a small manufacturer of golf equipment and supplies whose management has decided to move into the market for medium- and high-priced golf bags. Par, Inc.'s distributor has agreed to buy all the golf bags Par, Inc., produces over the next three months.

Each golf bag produced will require the following operations:

1. Cutting and dyeing the material
2. Sewing
3. Finishing (inserting umbrella holder, club separators, etc.)
4. Inspection and packaging

Par Inc. (linear case)

This production information is summarized in this table:



	Production Time (hours)	
Department	**Standard Bag**	**Deluxe Bag**
Cutting and Dyeing	7/10	1
Sewing	1/2	5/6
Finishing	1	2/3
Inspection and Packaging	1/10	1/4

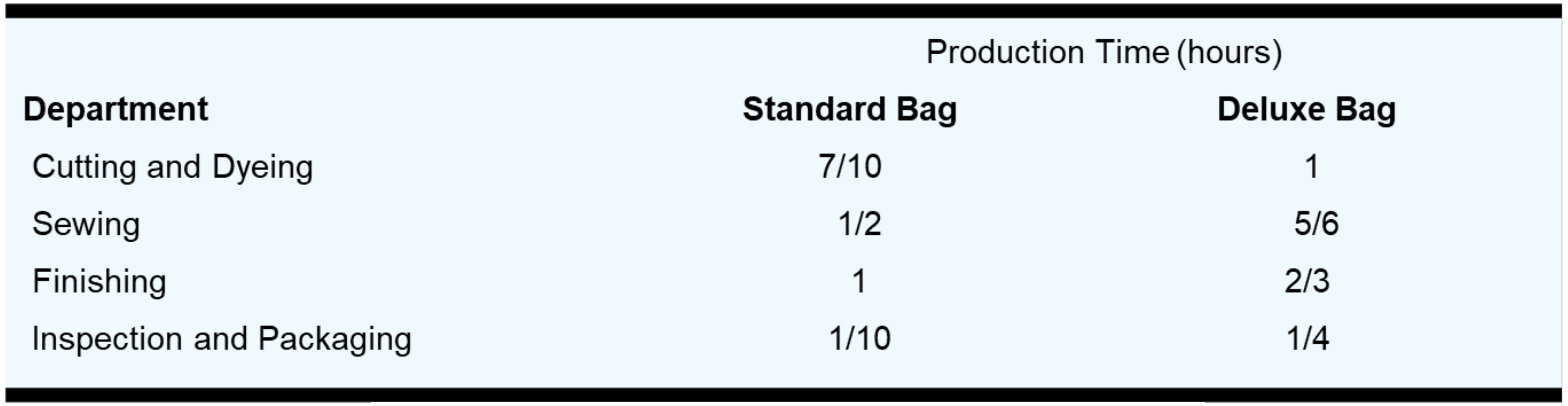
Par Inc. (linear case)

Par, Inc.'s production is constrained by a limited number of hours available in each department. The director of manufacturing estimates that 630 hours for cutting and dyeing, 600 hours for sewing, 708 hours for finishing, and 135 hours for inspection and packaging will be available for the production of golf bags during the next three months.

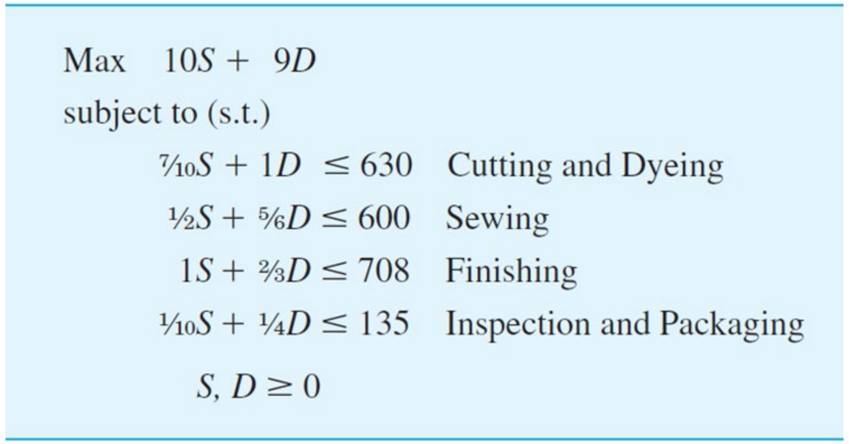
The accounting department analyzed the production data and arrived at prices for both bags that will result in a profit contribution1 of $10 for every standard bag and $9 for every deluxe bag produced.

Par Inc. (linear case)

The complete model for the Par, Inc., problem is as follows:



	Production Time (hours)	
Department	**Standard Bag**	**Deluxe Bag**
Cutting and Dyeing	7/10	1
Sewing	1/2	5/6
Finishing	1	2/3
Inspection and Packaging	1/10	1/4



Max $10S + 9D$

subject to (s.t.)

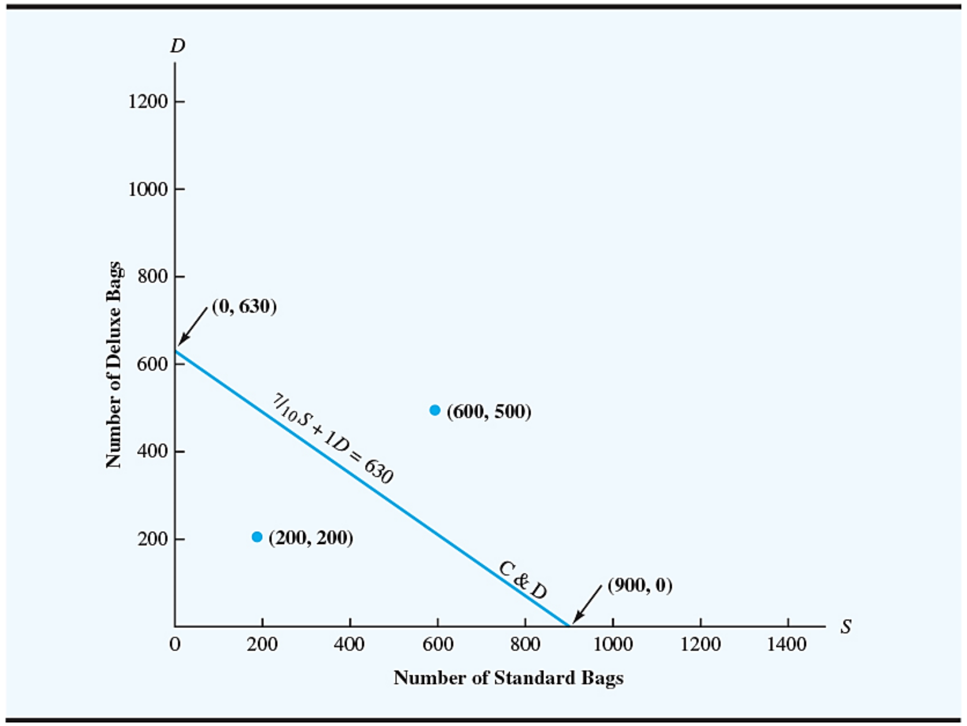
$$
\begin{aligned}
\tfrac{7}{10}S + 1D &\le 630 &&\text{Cutting and Dyeing}\\
\tfrac{1}{2}S + \tfrac{5}{6}D &\le 600 &&\text{Sewing}\\
1S + \tfrac{2}{3}D &\le 708 &&\text{Finishing}\\
\tfrac{1}{10}S + \tfrac{1}{4}D &\le 135 &&\text{Inspection and Packaging}\\
S, D &\ge 0
\end{aligned}
$$

Par Inc. (linear case)

Earlier, we saw that the inequality representing the cutting and dyeing constraint is:

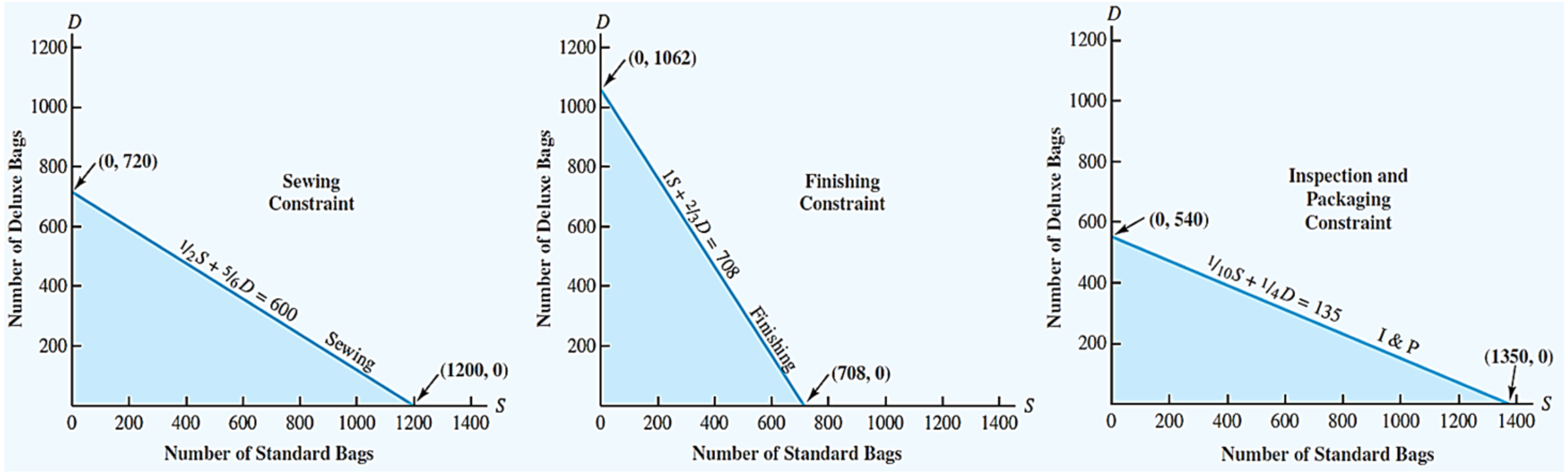
To show all solution points that satisfy this relationship, we start by graphing the solution points satisfying the constraint as an equality.

$$\frac{7}{10}S + 1D \leq 630$$



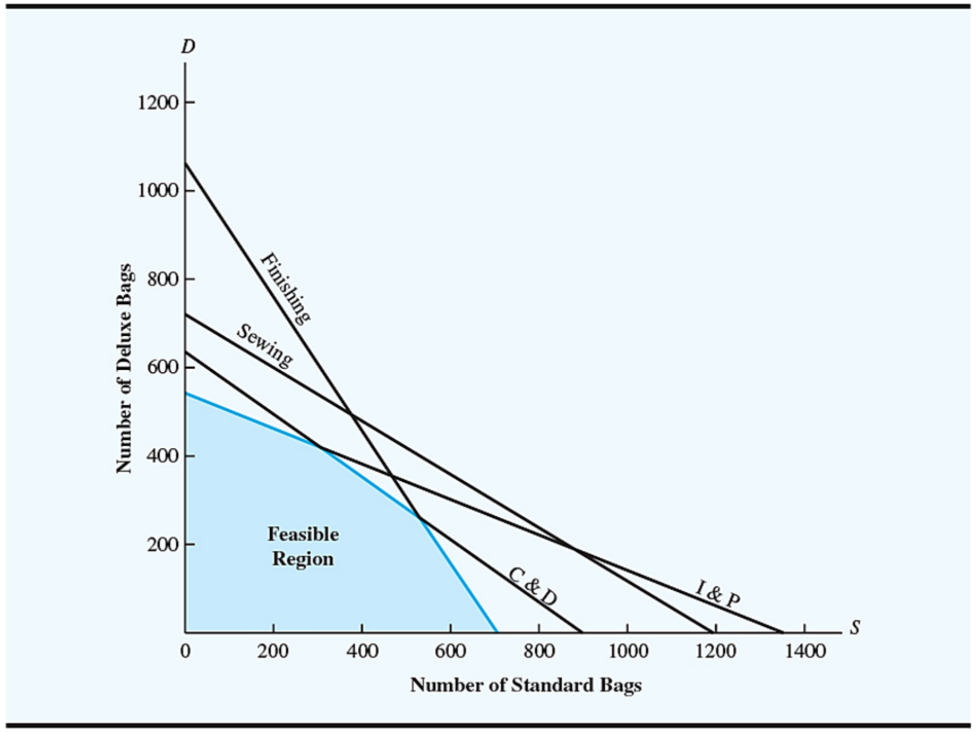
Par Inc. (linear case)

We continue by identifying the solution points satisfying each of the other three constraints.



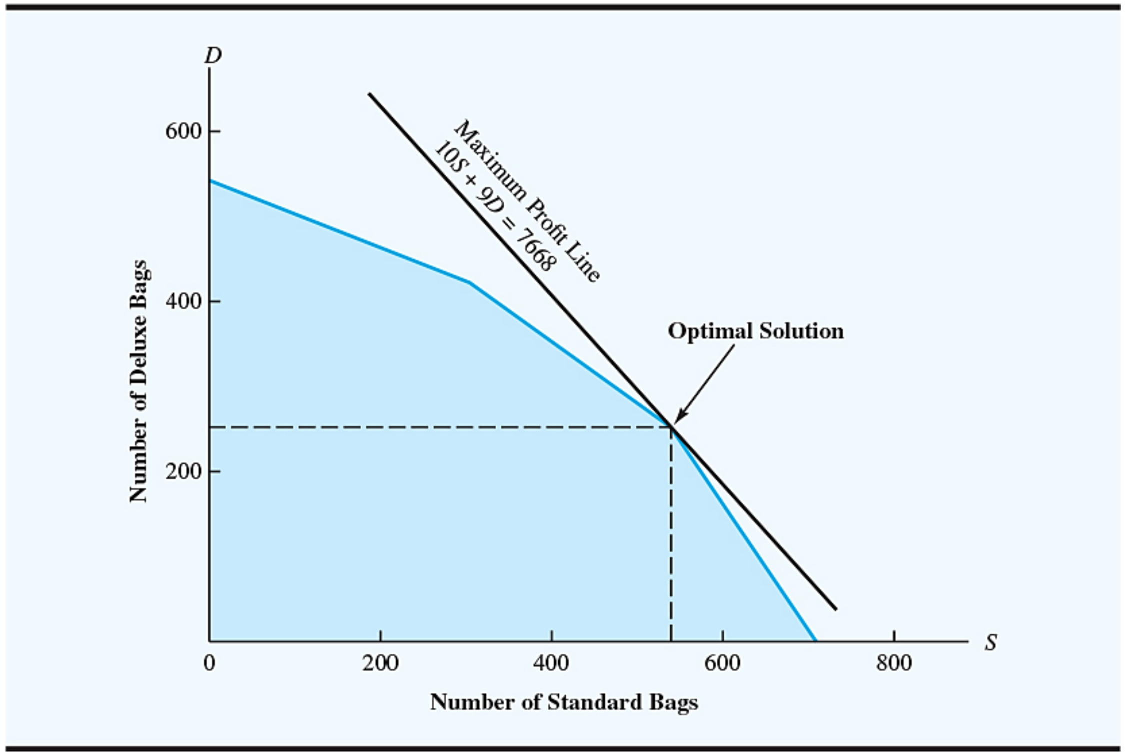
Graphical Solution

The graph shown identifies the feasible region:



Graphical Solution

The optimal solution point is at the intersection of the cutting and dyeing and the finishing constraint lines.



Optimal Solution (Linear case)

The optimal values of the decision variables *S* and *D* must satisfy dyeing and the finishing constraints simultaneously.

$$\frac{7}{10}S + 1D = 630$$ Dyeing Constraint

$$1S + \frac{2}{3}D = 708$$ Finishing constraint

This system of equations can be solved using substitution.

The exact location of the optimal solution point is *S* = 540 and *D* = 252. The optimal production quantities for Par, Inc., are 540 standard bags and 252 deluxe bags, with a resulting profit contribution of 10(540) + 9(252) = $7,668.

A Production Application – Par, Inc (nonlinear case)

An Unconstrained Problem

Par, Inc. has decided to move into the market for medium- and high-priced golf bags; Par's distributor has agreed to buy all the golf bags Par produces over the next three months

We introduce constrained and unconstrained nonlinear optimization problems by considering an extension of the Par, Inc. linear program

Consider the case in which the relationship between price and quantity sold causes the objective function to be nonlinear; the result is an unconstrained nonlinear program

A Production Application – Par, Inc

In formulating the linear programming model for the Par, Inc. problem, we assumed that the company could sell all of the Standard and Deluxe bags it could produce; depending on the price of the golf bags, this assumption may not hold

An inverse relationship usually exists between price and demand: as price increases, the quantity demanded decreases

Let P_S denote the price Par, Inc. charges for each Standard bag and P_D denotes the price for each Deluxe bag

A Production Application – Par, Inc

Assume that the demand for Standard bags, *S*, and the demand for Deluxe bags, *D*, are given by:

$$S = 2250 - 15P_s$$
$$D = 1500 - 5P_D$$

The cost to produce a Standard bag is $70 and the cost to produce each Deluxe golf bag is $150

A Production Application – Par, Inc

We can solve the equation, $S = 2250 - 15P_s$ for P_s

to show how the price of a standard bag is related to the number of standard bags sold:

$$P_s = 150 - (1/15)S.$$

Remember that the profit contribution for producing and selling*S* standard bags (revenue – cost) is

$$P_s S - 70S.$$

Substituting, gives the profit contribution for standard bags:

$$P_s S - 70S$$

$$= (150 - (1/15)S)S - 70S = 80S - \frac{1}{15}S^2$$

A Production Application – Par, Inc

Using the same logic we used to develop the profit contribution for standard bags, for the deluxe bags we have:

$$
\begin{aligned}
D &= 1500 - 5P_D \\
P_D &= 300 - \frac{1}{5}D \\
\text{Profit} &= \text{Revenue - Cost} \\
&= P_D D - 150D \\
&= (300 - \frac{1}{5}D)D - 150D \\
&= 150D - \frac{1}{5}D^2
\end{aligned}
$$

A Production Application – Par, Inc

Total profit contribution is the sum of the profit contribution for standard bags and the profit contribution for deluxe bags. Thus, total profit contribution is written as:

$$\text{Total Profit} = 80S - \frac{1}{15}S^2 + 150D - \frac{1}{5}D^2$$

If we want to maximize profit then the above formula will be our objective function that is a nonlinear function of number of standard bags and deluxe bags.

This is a special case of nonlinearity where the nonlinearity is second order. This is called quadratic programming.

A Production Application – Par, Inc., Revisited

Using a computer solution method (CVXOPT package, see solution in the following slides), we find that the values of *S* and *D* that maximize the profit contribution function are *S* = 600 and *D* = 375.

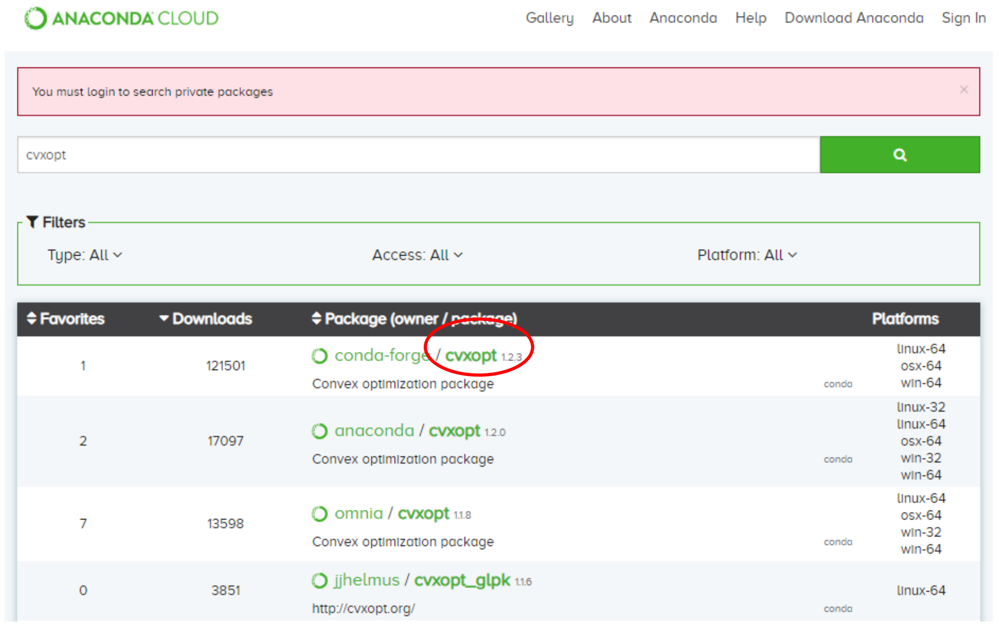
The corresponding prices are \$110 for standard bags and \$225 for deluxe bags, and the profit contribution is \$52,125.

These values provide the optimal solution for Par, Inc., if all production constraints are also satisfied.

Tools (Convex Optimization Python Library)

A common Python library used for solving convex optimization is CVXOPT that is available through Anaconda distribution.

The following are the screenshots of the library and the command to install it in your optimization environment.



You must login to search private packages

cvxopt

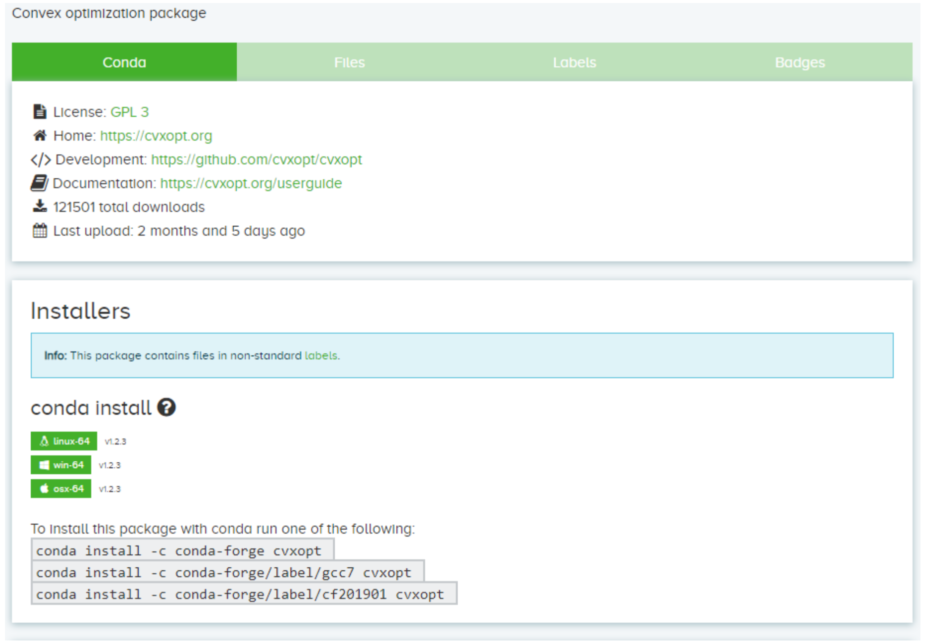
Filters

Type: All

Access: All

Platform: All

Favorites	Downloads	Package (owner / package)		Platforms
1	121501	conda-forge / cvxopt 1.2.3 Convex optimization package	conda	linux-64 osx-64 win-64
2	17097	anaconda / cvxopt 1.2.0 Convex optimization package	conda	linux-32 linux-64 osx-64 win-32 win-64
7	13598	omnia / cvxopt 1.1.8 Convex optimization package	conda	linux-64 osx-64 win-32 win-64
0	3851	jjhelmus / cvxopt_glpk 1.1.6 http://cvxopt.org/	conda	linux-64



Convex optimization package

License: GPL 3
Home: https://cvxopt.org
Development: https://github.com/cvxopt/cvxopt
Documentation: https://cvxopt.org/userguide
121501 total downloads
Last upload: 2 months and 5 days ago

Installers

Info: This package contains files in non-standard labels.

conda install

linux-64 v1.2.3
win-64 v1.2.3
osx-64 v1.2.3

To install this package with conda run one of the following:

```
conda install -c conda-forge cvxopt
conda install -c conda-forge/label/gcc7 cvxopt
conda install -c conda-forge/label/cf201901 cvxopt
```

Quadratic Programming Using CVXOPT

The CVXOPT package requires that we formulate the problem as a system of equations.

A simple example using CVXOPT package:

Assume the following objective function and constraints:

$$\text{Obj function} = \frac{1}{2}x^2 + 3x + 4y$$

subject to (s.t.)

$$x, y \geq 0$$

$$x + 3y \geq 15$$

$$2x + 5y \leq 100$$

$$3x + 4y \leq 80$$

Quadratic Programming Using CVXOPT

Converting the problem into standard form:

The standard form of CVXOPT requires the following format:

$$\text{obj fnc} = \frac{1}{2} x^T P\, x + q\, x, \text{ where P is a positive semidefinite matrix, i.e. } |P| \geq 0$$

$$s.t.$$

$$G\, x \leq h \ \ \text{inequality constraints}$$

$$A\, x = b \ \ \text{equality constraints}$$

This means that other formats needs to conform/or be converted to the standard form. For example if there is condition x,y >= 0 should be converted to –x, -y <= 0 since every inequality should be written as less than or equal to.

If a term is missing in the formulation of objective function it should still be replaced with a zero entry.

Quadratic Programming Using CVXOPT

Formulation of the example in the standard form:

$$\text{Obj function} = \frac{1}{2}x^2 + 3x + 4y$$

subject to (s.t.)

$$x, y \geq 0$$

$$x + 3y \geq 15$$

$$2x + 5y \leq 100$$

$$3x + 4y \leq 80$$

Non-standard form

$$\min_{x,y} \frac{1}{2}\begin{bmatrix} x \\ y \end{bmatrix}^T \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 3 \\ 4 \end{bmatrix}^T \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} -1 & 0 \\ 0 & -1 \\ -1 & -3 \\ 2 & 5 \\ 3 & 4 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ -15 \\ 100 \\ 80 \end{bmatrix}$$

$A, b = 0$ since there is no equality constraint

Standard form

CVXOPT Python code

```
# Import the necessary packages
import numpy
from cvxopt import matrix
from cvxopt import solvers

# Define QP parameters (directly)
P = matrix([[1.0,0.0],[0.0,0.0]])
q = matrix([3.0,4.0])
G = matrix([[-1.0,0.0,-1.0,2.0,3.0],[0.0,-1.0,-3.0,5.0,4.0]])
h = matrix([0.0,0.0,-15.0,100.0,80.0])

# Define QP parameters (with NumPy) uncomment if you want to use the numpy implementation
#P = matrix(numpy.diag([1,0]), tc='d’)
#q = matrix(numpy.array([3,4]), tc='d’)
#G = matrix(numpy.array([[-1,0],[0,-1],[-1,-3],[2,5],[3,4]]), tc='d’)
#h = matrix(numpy.array([0,0,-15,100,80]), tc='d')

# Construct the QP, invoke solver
sol = solvers.qp(P,q,G,h)
# Extract optimal value and solution
print(sol['x']) # [7.13e-07, 5.00e+00]
print(sol['primal objective’]) # 20.0000061731
```

CVXOPT Python code Explanation

Note how we collapsed all inequality constraints into a single G matrix of the standard form.

Since there are no equality constraints, we do not need to provide the empty A; b.

Note that even though y^2 did not appear in the original objective, we had to include it with zero coefficients in P because the solver parameters must be defined using the full set of variables.

Even if certain variables only appear in constraints, they will still need to be expressedwith

zero coefficients in the objective parameters, and vice versa.

The CVXOPT library requires that all entries in the matrices be double so if you enter them as integers you could end up with cryptic errors. An alternate solution and variable definition has been presented using Numpy. The Numpy implementation has been commented out. Try it both ways and observe that both implementations produce the same results. The parameter tc=d is a way of defining the coefficients as double. This form can also be used in the direct implementation.

CVXOPT Implementation of Par. Inc

```
import numpy as np
from cvxopt import matrix
from cvxopt import solvers

# Define QP parameters (use Numpy). pay attention how we have changed the sign of the objective function to keep P
# positive semi-definite

P = matrix(np.array([[2/15., 0.], [0., 2/5.]]),tc=‘d’)
q = matrix(np.array([-80., -150.]).reshape((2,)), tc='d’)
G = matrix(np.array([[0.7, 1.], [0.5, 5/6.], [1., 2/3.], [0.1, 0.25]]), tc='d’
h = matrix(np.array([630., 600., 708., 135.]).reshape((4,)), tc='d’)

# Construct the QP, invoke solver, If we let G and h out we are solving the unconstrained optimization
# since we are stating that G and h are None or not provided


sol = solvers.qp(P,q)

# Extract optimal value and solution

print(sol['x']) # unconstrained solution S = 600, D = 375

sol['primal objective’] # unconstrained optimal solution = 52125.0
```

Unconstrained Optimization Issue

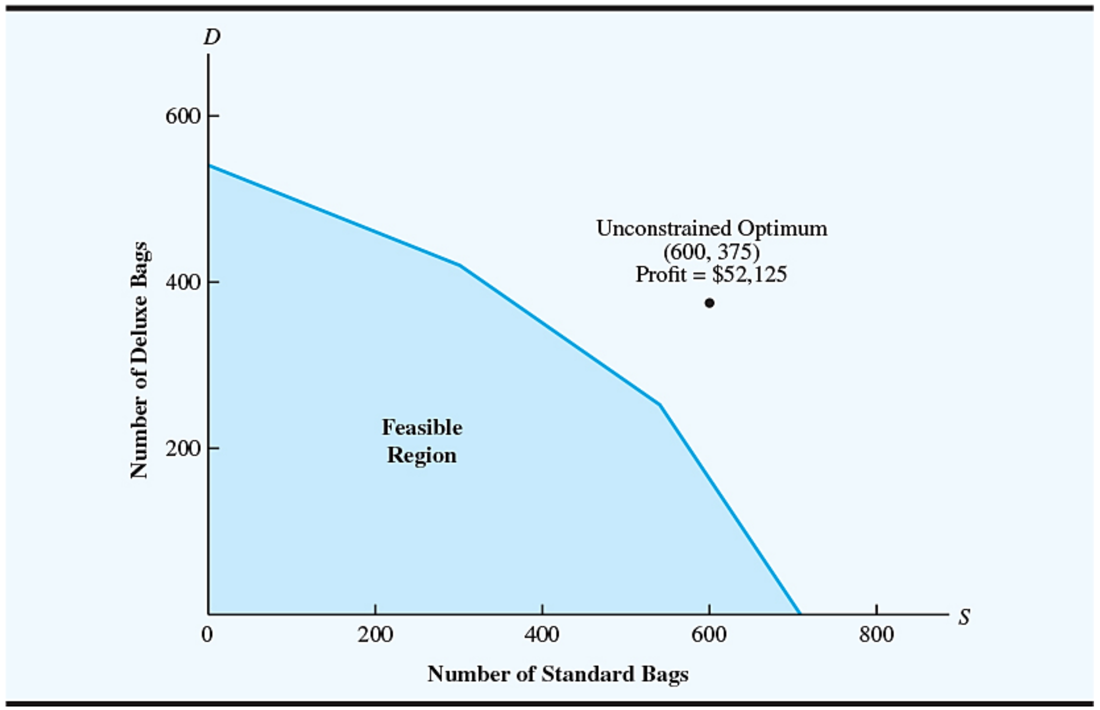
Unfortunately, Par, Inc., cannot make the profit contribution associated with the optimal solution to the unconstrained problem because the constraints defining the feasible region are violated. For instance, the cutting and dyeing constraint is

$$7/10S + D \leq 630$$

A production quantity of 600 standard bags and 375 deluxe bags will require 7/10(600) + 1(375) = 795 hours, which exceeds the limit of 630 hours by 165 hours.

Unconstrained Optimization Issue

The feasible region for the original Par, Inc., problem along with the unconstrained optimal solution point (600, 375) is shown here:



The unconstrained optimum of (600, 375) is outside the feasible region.

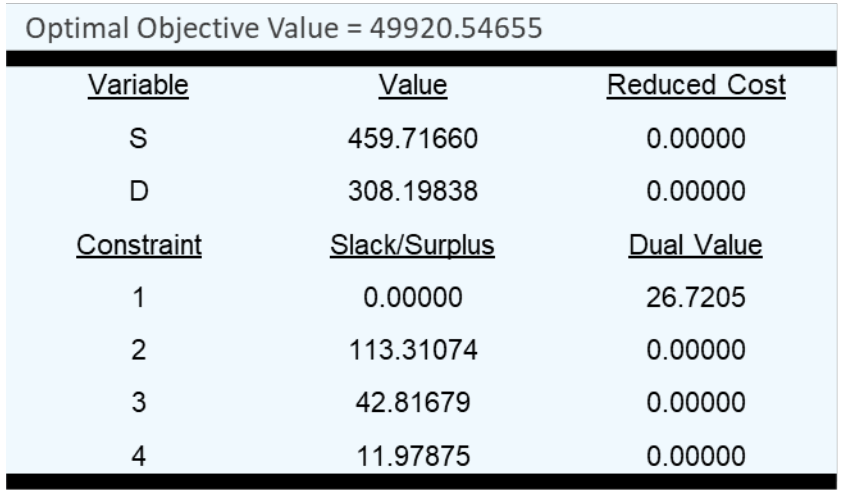
A Constrained Problem

The complete mathematical model for the Par, Inc., constrained nonlinear maximization problem is:

$$
\begin{array}{lrl}
\text{Max} & \multicolumn{2}{l}{80S - \tfrac{1}{15}S^2 + 150D - \tfrac{1}{5}D^2} \\
\text{s.t.} & & \\
& \tfrac{7}{10}S + D \leq 630 & \text{Cutting and Dyeing} \\
& \tfrac{1}{2}S + \tfrac{5}{6}D \leq 600 & \text{Sewing} \\
& S + \tfrac{2}{3}D \leq 708 & \text{Finishing} \\
& \tfrac{1}{10}S + \tfrac{1}{4}D \leq 135 & \text{Inspection and Packaging} \\
& S, D \geq 0 &
\end{array}
$$

Constrained Problem Solution

The optimal value of the objective function is $49,920.55. The optimal solution is to produce 459.7166 standard bags and 308.1984 deluxe bags. In the Slack/Surplus column, the value of 0 in Constraint 1 means that the optimal solution uses all the labor hours in the cutting and dyeing department; but the nonzero values in rows 2–4 indicate that slack hours are available in the other departments.



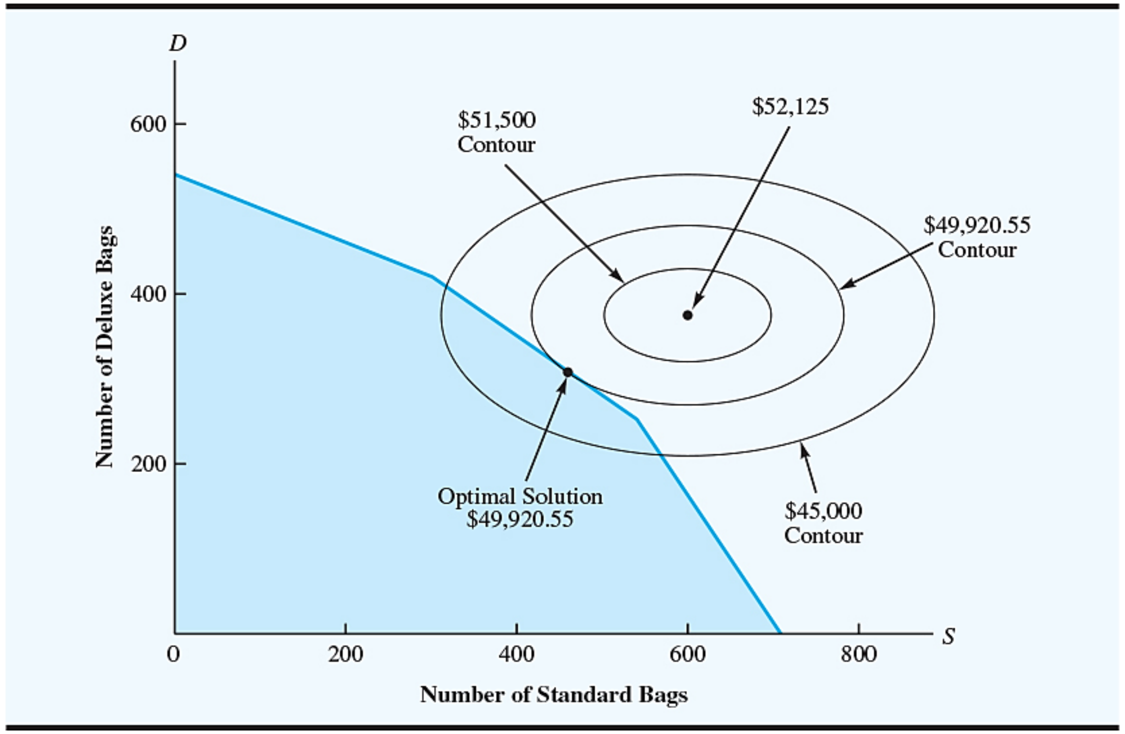
Optimal Objective Value = 49920.54655

Variable	Value	Reduced Cost
S	459.71660	0.00000
D	308.19838	0.00000
Constraint	**Slack/Surplus**	**Dual Value**
1	0.00000	26.7205
2	113.31074	0.00000
3	42.81679	0.00000
4	11.97875	0.00000

Constrained Problem Solution

Here we see three profit contribution *contour lines*. Each point on the same contour line is a point of equal profit. Here, the contour lines show profit contributions of $45,000, $49,920.55, and $51,500.

For the Par, Inc., problem with a quadratic objective function, the profit contours are ellipses.



Local vs. Global Optimum

A feasible solution is a local optimum if there are no other feasible solutions with a better objective function value in the immediate neighborhood.

- For a maximization problem the local optimum corresponds to a local maximum.
- For a minimization problem the local optimum corresponds to a local minimum.

A feasible solution is a global optimum if there are no other feasible points with a better objective function value in the feasible region.

A global optimum is also a local optimum

Multiple Local Optima

Nonlinear optimization problems can have multiple local optimal solutions, in which case we want to find the best local optimum.

Nonlinear problems with multiple local optima are difficult to solve and pose a serious challenge for optimization software.

In these cases, the software can get “stuck” and terminate at a local optimum.

There can be a severe penalty for finding a local optimum that is not a global optimum.

Developing algorithms capable of finding the global optimum is currently a very active research area.

Single Local Maximum

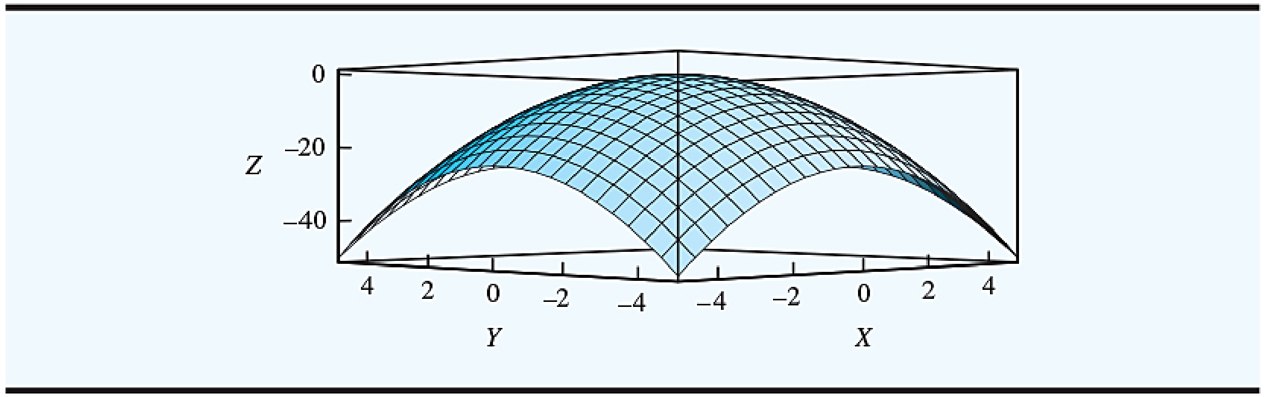
Consider the function $f(X, Y) = -X^2 - Y^2$.

A function that is bowl-shaped down is a <u>concave function</u>.

The maximum value for this particular function is 0 and the point (0, 0) gives the optimal value of 0.

Functions such as this one have a single local maximum that is also a global maximum.

This type of nonlinear problem is relatively easy to maximize.



Single Local Minimum

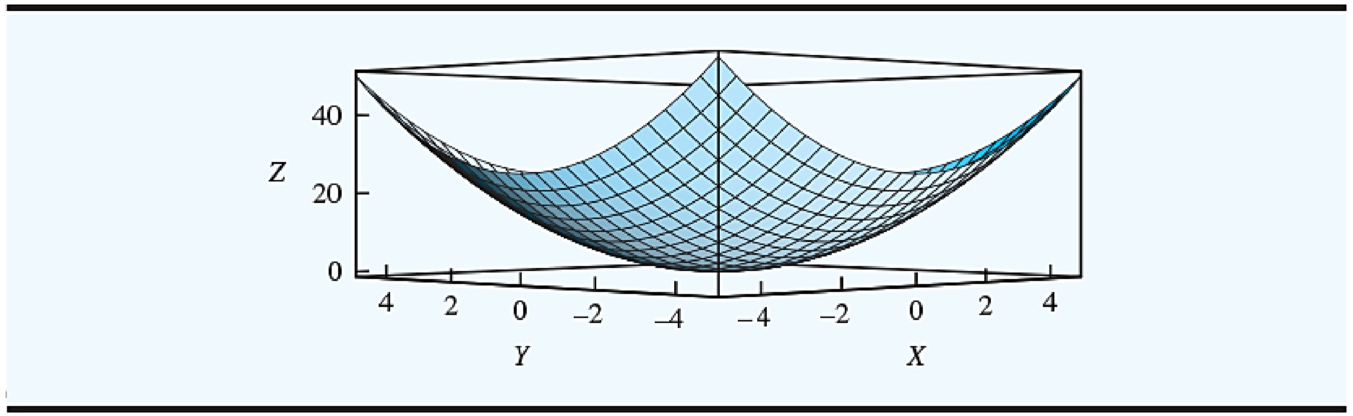
Consider the function $f(X, Y) = X^2 + Y^2$.

A function that is bowl-shaped up is a convex function.

The minimum value for this particular function is 0 and the point (0, 0) gives the optimal value of 0.

Functions such as this one have a single local minimum that is also a global minimum.

This type of nonlinear problem is relatively easy to minimize.

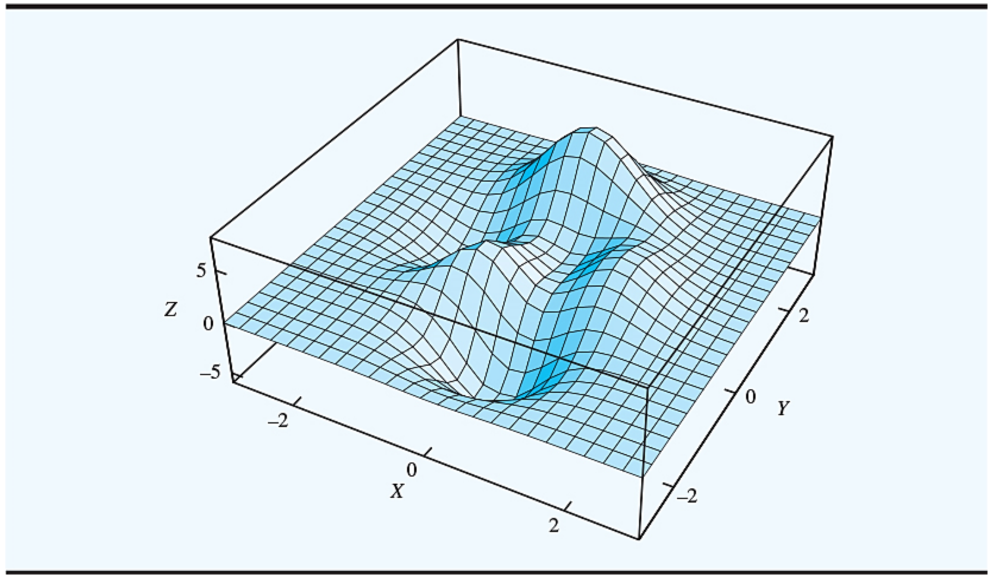


Multiple Local Optima

Consider the function

$$f(X,Y) = 3(1-X)^2 e^{\left(-X^2-(Y+1)^2\right)} - 10\left(X/5 - X^3 - Y^5\right)e^{\left(-X^2-Y^2\right)} - e^{\left(-(X+1)^2-Y^2\right)}/3$$

The hills and valleys in the graph show that this function has several local maximums and local minimums.
There are two local minimums, one of which is the the global minimum.
There are three local maximums, one of which is the global maximum



Dual Values

Recall that the dual value is the change in the value of the optimal solution per unit increase in the right-hand side of the constraint.

The interpretation of the dual value for nonlinear models is exactly the same as it is for LPs.

However, for nonlinear problems the allowable increase and decrease are not usually reported.

This is because for typical nonlinear problems the allowable increase and decrease are zero.

That is, if you change the right-hand side by even a small amount, the dual value changes.

Markowitz Portfolio Model (Quadratic programming example)

There is a key tradeoff in most portfolio optimization models between risk and return.

In the index fund model this tradeoff is managed passively.

The Markowitz mean-variance portfolio model provides a very convenient way for an investor to actively trade-off risk versus return.

We will now demonstrate the Markowitz portfolio model byproviding an example.

Markowitz Portfolio Model

There are two basic ways to formulate the Markowitz model:

(1) Minimize the variance of the portfolio subject to constraints on the expected return, and

(2) Maximize the expected return of the portfolio subject to a constraint on risk.

We will now demonstrate the first (1) formulation, assuming the client requires the expected portfolio return to be at least 10 percent.

Markowitz Portfolio Model

The portfolio variance is the average of the sum of the squares of the deviations from the mean value under each scenario.

The larger the variance value, the more widely dispersed the scenario returns are about the average return value.

If the portfolio variance were equal to zero, then every scenario return R_i would be equal.

In most portfolio optimization models:

- The return used is the expected (or average) return of the possible outcomes
- The risk is some measure of variability in these possible outcomes

Markowitz Portfolio Model Example

Illustration: Hauck Investment Services

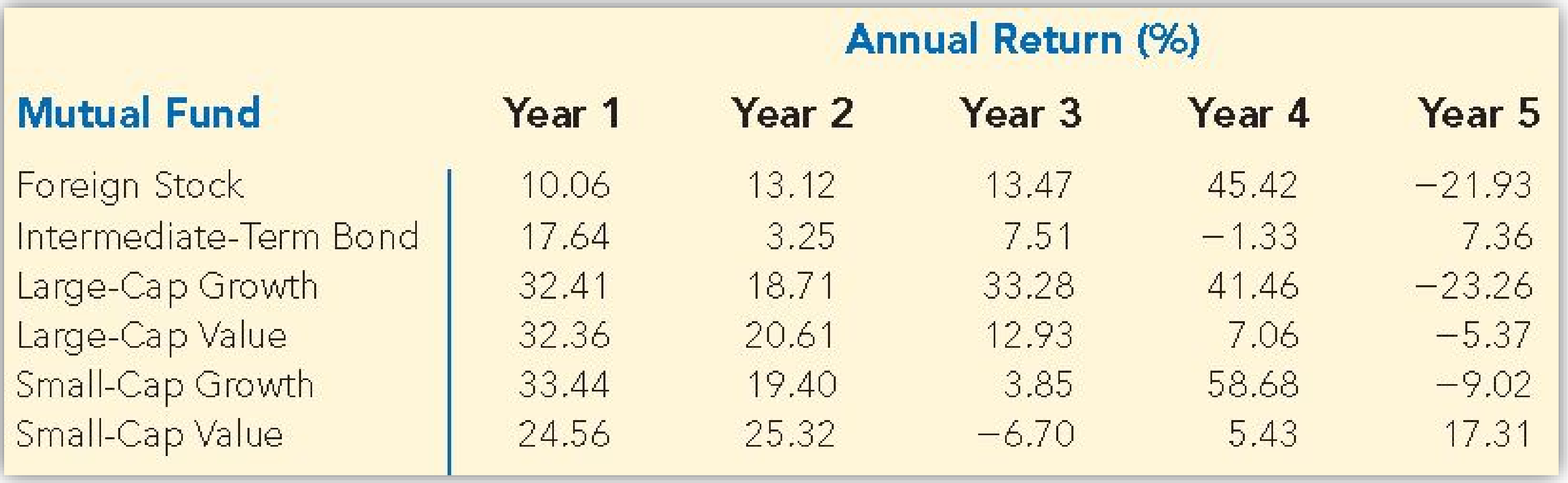
Hauck Investment Services designs annuities, IRAs, 401(k) plans, and other investment vehicles for investors with a variety of risk tolerances

Hauck would like to develop a portfolio model that can be used to determine an optimal portfolio involving a mix of six mutual funds

Table 1 shows the annual return (percent) for five 1-year periods for the six mutual funds

Markowitz Portfolio Model Example

Table 1: Mutual Fund Performances in Five Selected Years (Used as Planning Scenarios for the Next 12 Months)



	Annual Return (%)				
Mutual Fund	**Year 1**	**Year 2**	**Year 3**	**Year 4**	**Year 5**
Foreign Stock	10.06	13.12	13.47	45.42	−21.93
Intermediate-Term Bond	17.64	3.25	7.51	−1.33	7.36
Large-Cap Growth	32.41	18.71	33.28	41.46	−23.26
Large-Cap Value	32.36	20.61	12.93	7.06	−5.37
Small-Cap Growth	33.44	19.40	3.85	58.68	−9.02
Small-Cap Value	24.56	25.32	−6.70	5.43	17.31

Markowitz Portfolio Model Example

The following decision variables:

FS = Proportion of portfolio invested in the foreign stock mutual fund

IB= Proportion of portfolio invested in the intermediate-term bond fund

LG= Proportion of portfolio invested in the large-cap growth fund

LV= Proportion of portfolio invested in the large-cap value fund

SG= Proportion of portfolio invested in the small-cap growth fund

SV= Proportion of portfolio invested in the small-cap value fund

Markowitz Portfolio Model Example

$$FS + IB + LG + LV + SG + SV = 1$$

Let R_1 denote the portfolio return if the scenario represented by year 1 occurs

Let R_2 denote the portfolio return if the scenario represented by year 2 occurs, and so on

If p_s is the probability of scenario *s*, among *n* possible scenarios, then the *expected* return for the portfolio is *R* where:

$$\bar{R} = \sum_{s=1}^{n} p_s R_s$$

Markowitz Portfolio Model Example

If we assume that the five planning scenarios in the Hauck Financial Services model are equally likely to occur

$$\bar{R} = \sum_{s=1}^{5} {}^{1}/_{5}\, R_s = {}^{1}/_{5} \sum_{s=1}^{5} R_s$$

The measure of risk most often associated with the Markowitz portfolio model is the variance of the portfolio’s return

$$Var = \sum_{s=1}^{n} p_s (R_s - \bar{R})^2$$

Markowitz Portfolio Model Example

For the Hauck Financial Services example, the five planning scenarios are equally likely, thus:

$$Var = \sum_{s=1}^{n} {}^{1}\!/_{5}\,(R_s - \bar{R})^2 = {}^{1}\!/_{5} \sum_{s=1}^{n} (R_s - \bar{R})^2$$

Assume that Hauck clients would like to construct a portfolio from the six mutual funds listed in Table 13.2 that will minimize their risk as measured by the portfolio variance

The complete Markowitz model involves 12 variables and 8 constraints (excluding the nonnegativity constraints)

$$\text{Min } {}^{1}\!/_{5} \textstyle\sum_{s=1}^{n} p_s (R_s - \bar{R})^2$$

Markowitz Portfolio Model Example

$$\text{Min } \tfrac{1}{5}\sum_{s=1}^{5}(R_S - \overline{R})^2$$

s.t.

$$\begin{array}{rcrcrcrcrcrcl}
10.06FS & + & 17.64IB & + & 32.41LG & + & 32.36LV & + & 33.44SG & + & 24.56SV & = & R_1 \\
13.12FS & + & 3.25IB & + & 18.71LG & + & 20.61LV & + & 19.40SG & + & 25.32SV & = & R_2 \\
13.47FS & + & 7.51IB & + & 33.28LG & + & 12.93LV & + & 3.85SG & - & 6.70SV & = & R_3 \\
45.42FS & - & 1.33IB & + & 41.46LG & + & 7.06LV & + & 58.68SG & + & 5.43SV & = & R_4 \\
-21.93FS & + & 7.36IB & - & 23.26LG & - & 5.37LV & - & 9.02SG & + & 17.31SV & = & R_5 \\
FS & + & IB & + & LG & + & LV & + & SG & + & SV & = & 1 \\
 & & & & & & & & & & \tfrac{1}{5}\sum_{s=1}^{5}R_S & = & \overline{R} \\
 & & & & & & & & & & \overline{R} & \geq & 10
\end{array}$$

$$FS, IB, LG, LV, SG, SV \geq 0$$

The objective for the Markowitz model is to minimize portfolio variance

The solution for this model using a required return of at least 10 percent appears in Figure 13.11

Markowitz Portfolio Example Solution

The minimum value for the portfolio variance is 27.136

This solution implies that the clients will get an expected return of 10 percent and minimize their risk as measured by portfolio variance by investing approximately

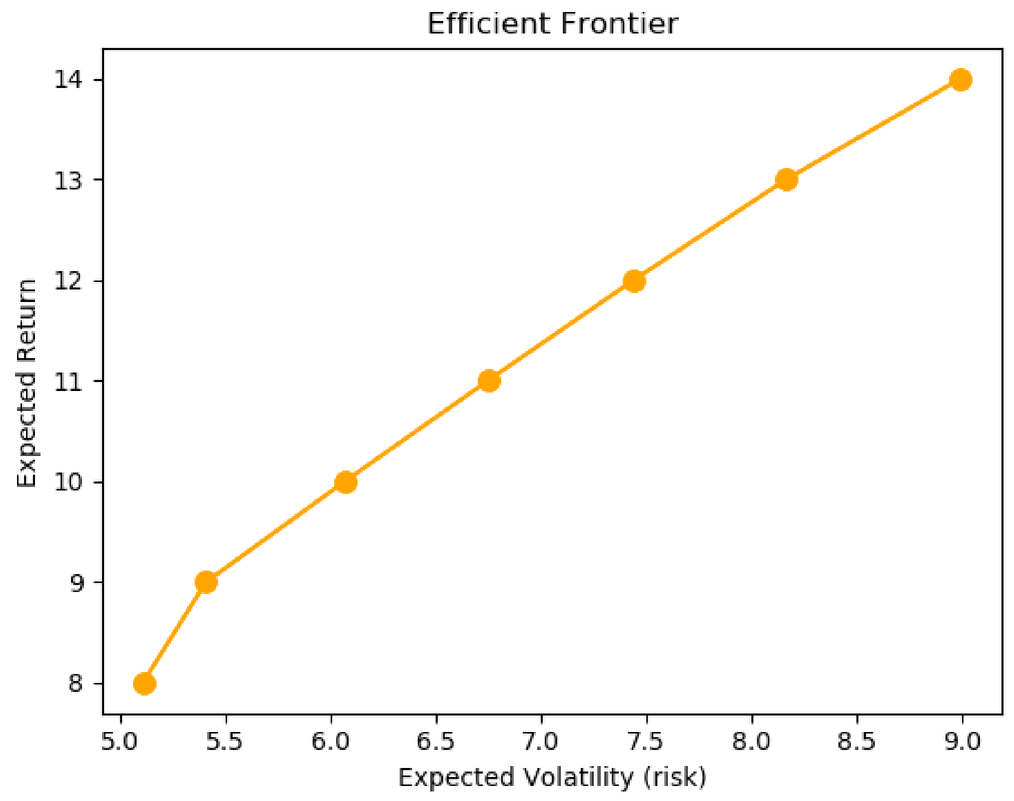
- 16 percent of the portfolio in the foreign stock fund (FS = 0.158)
- 53 percent in the intermediate bond fund (IB =0.525)
- 4 percent in the large-cap growth fund (LG = 0.042)
- 27 percent in the small-cap value fund (SV =0.274)

Markowitz Portfolio Example Solution

The figure below is a graph of the minimum portfolio variances versus required expected returns as required expected return is varied from 8 percent to 12 percent in increments of 1 percent

This graph is called the **efficient frontier**; each point on the efficient frontier is the minimum possible risk (measured by portfolio variance) for the given return

By looking at the graph, investors can select the mean-variance combination with which they are most comfortable



Markowitz Example Solution with CVXOPT

```
# Import the necessary packages
import numpy as np
from cvxopt import matrix
from cvxopt import solvers
import matplotlib.pyplot as plt
#import matplotlib as mpl
#import seaborn as sns


def calculate_MPT(returns,r_min):
 # calculation of P matrix in the quadratic equation 1/2 xt P x
 cov = np.matrix(np.cov(returns.T))
# number of stocks or assessts
N = returns.shape[1]
  # R-bar is the average returns from data. Would be compared to the minimum returns
 rbar = np.matrix(returns.mean(0))
 # define list of optimal / desired mus for which we'd like to find the optimal sigmas
 # This is the minimum returns plus accumulation of returns over 5 years. you can
 # also make all 5 years minimum returns equal to 10
 optimal_mus = []
 for i in range(N):
       optimal_mus.append(r_min)
       r_min += (rbar.mean() / 100)
```

Markowitz Example Solution with CVXOPT

```
# calculate average minimum expected portfolio returns
    mu = np.array(optimal_mus).mean()
    # constraint matrices for quadratic programming. Here I used  cvxopt package instead of quadprog that seems to be easier to use
    # parameters of cvxopt are similar to quadprog
    P = matrix(cov)
    q = matrix(np.zeros((N, 1)))
    G = matrix(np.concatenate((-np.array(rbar), -np.identity(N)), 0))
    h = matrix(np.concatenate((-np.ones((1,1))*mu, np.zeros((N,1))),0))
    A = matrix(1.0, (1,N))
    b = matrix(1.0)

    # hide optimization
    solvers.options['show_progress'] = False

    # calculate portfolio weights, every weight vector is of size Nx1
    # find optimal weights with qp(P, q, G, h, A, b)
    optimal_weights = solvers.qp(P, q, G, h, A, b)
    w = optimal_weights['x']
```

Markowitz Example Solution with CVXOPT

```
    # find optimal sigma
       # \sigma = w^T * Cov * w
       #optimal_sigmas = [np.sqrt(np.matrix(w).T * cov.T.dot(np.matrix(w)))[0,0] for w in optimal_weights]
       optimal_sigma = np.sqrt(np.matrix(w).T * cov.T.dot(np.matrix(w)))

       return w, optimal_sigma.item(0), mu
    # print optimum weights
    def main():
       # define returns, returns are a matrix TxN. N is the number of assessts/stocks,
      # T number of years, time in time series
       returns = np.array([[10.06,17.64,32.41,32.36,33.44,24.56],
                      [13.12,3.25,18.71,20.61,19.40,25.32],
                      [13.47,7.51,33.28,12.93,3.85,-6.7],
                      [45.42,-1.33,41.46,7.06,58.68,5.43],
                      [-21.93,7.36,-23.26,-5.37,-9.02,17.31]])
       minReturns = [8, 9, 10, 11, 12, 13, 14]
       mus = []
       optimal_sigmas = []
       w = []
       for i in range(len(minReturns)):
          weights, optimal_sig, mu =calculate_MPT(returns,minReturns[i])
          mus.append(mu)
          optimal_sigmas.append(optimal_sig)
          w.append(weights)
          print(weights)
       # plot Efficient Frontier
       plt.plot(optimal_sigmas, [x for x in minReturns],'y-o', color='orange', markersize=8, label='Efficient Frontier')
       plt.xlabel('Expected Volatility (risk)')
       plt.ylabel('Expected Return')
       plt.title('Efficient Frontier')

main()
```