AAE 706: Applied Risk Analysis

Nonconvexity

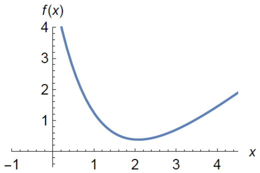
Thomas F. Rutherford

Department of Agricultural and Applied Economics
University of Wisconsin, Madison

17 April, 2023

Convex programs

- We have covered LP, QP and NLP
- Convex programs can be solved reliably and efficiently
- Optimal cost can be bounded above and below (duality)
- Local optima are global optima

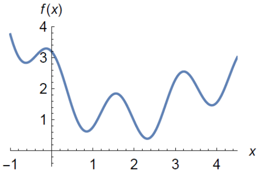


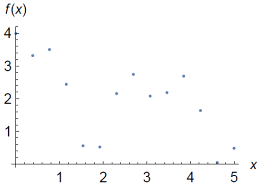
Outline

- Convex Programs
- Discrete and nonconvex models
- Mixed-integer programming
- A trivial example
- A nonlinear example

Nonconvex programs

- In general, cannot be efficiently solved
- Cost cannot be bounded easily
- Usually we can only guarantee local optimality
- Difficulty depends strongly on the instance





Topics in Nonconvex Programming

- Integer *linear* programs
  - It's an LP where some or all variables are discrete (boolean, integer, or general discrete-valued)
  - If all variables are integers, its called IP or ILP
  - If the model involves a mixture of continuous and discrete variables, it's called MIP or MILP
- *Nonconvex nonlinear* programs
  - If all variables are *continuous*, the problem class is NLP (but a *global* optimization code may be required).
  - If some variables are discrete, it's called MINLP, and a specialized solver for this type of model is required.
- Approximation and *relaxation*
  - Can we solve a convex problem instead?
  - If not, can we approximate?

Discrete variables

Why do we need discrete variables?

1. A decision variable may be fundamentally discrete.
   - The light is either on or off $\{1, 0\}$
   - Number of widgets produced $\{0, 1, 2, ...\}$
   - Bill amount $\{\$1, \$5, \$10, \$20, \$50, \$100\}$

Discrete variables

Why do we need discrete variables?

2. Discrete variables provide an *algebraic* representation of various logical conditions.
   - At most two of the three machines can run at once.

     $$z_1 + z_2 + z_3 \leq 2$$

     in which ($z_i = 1$ if machine i is running).
   - If machine 1 is running, so must machine 2.

     $$z_1 \leq z_2$$

Outline

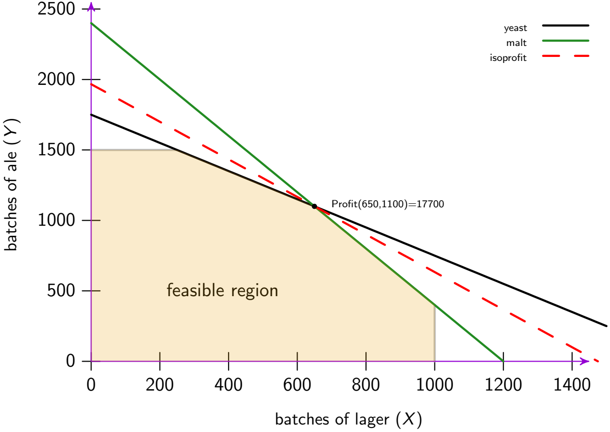
- Convex Programs
- Discrete and nonconvex models
- Mixed-integer programming
- A trivial example
- A nonlinear examples

Back to Making Beer

The local brewery produces two varieties of beer (lagers and ales) which are marketed in taverns and grocery around town. At the moment, they are planning production for fall. Each beer requires malt, hops and yeast. The lagers return \$120 in profit per batch while ales earn only \$90 per batch. Lagers are made with German hops, while ales are made with Wisconsin hops. There are currently sufficient German hops in stock for 1000 batches of lager and Wisconsin hops for 1500 batches of ale. Lager requires 4 kg of malt per batch while ale uses only 2 kg. Both beers require one kg of yeast per batch. There are 1,750 kg of yeast and 4800 kg of malt on hand.

What quantities of lager and ale should be produced from these supplies to maximize total profit assuming that all that are made can be sold?

Brewery Optimization



$$\max_{x,y} 120x + 90y$$

subject to:

$$4x + 2y \leq 4800$$

$$x + y \leq 1750$$

$$0 \leq x \leq 1000$$

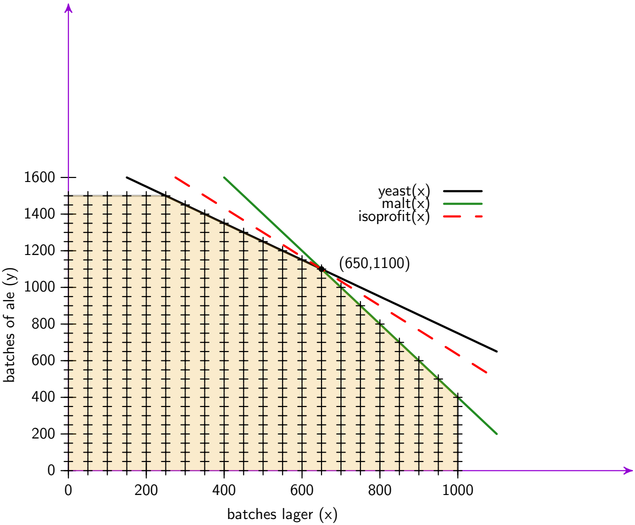
$$0 \leq y \leq 1500$$

In which:

x : batches of lager

y : batches of ale

Restrict Solution to 50 Batch Shipments



$$\max_{x,y} 120x + 90y$$

subject to:

$$4x + 2y \leq 4800$$

$$x + y \leq 1750$$

$$0 \leq x \leq 1000$$

$$0 \leq y \leq 1500$$

$$(x, y) \in \{0, 50, 100, 150, \ldots\}$$

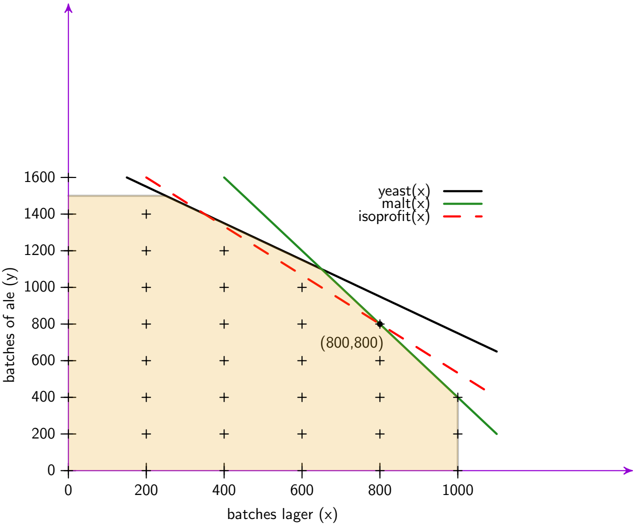
In which:

x : batches of lager

y : batches of ale

Same solution!

Restrict Solution to 200 Batch Orders



$$\max_{x,y} 120x + 90y$$

subject to:

$$4x + 2y \leq 4800$$

$$x + y \leq 1750$$

$$0 \leq x \leq 1000$$

$$0 \leq y \leq 1500$$

$$(x, y) \in \{0, 200, 400, 600, \ldots\}$$

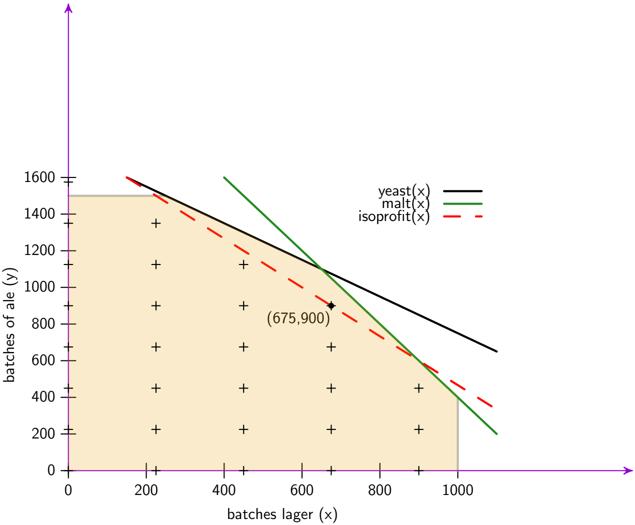
In which:

x : batches of lager

y : batches of ale

Non-vertex solution!

Restrict Solution to 225 Batch Shipments



$$\max_{x,y} 120x + 90y$$

subject to:

$$4x + 2y \leq 4800$$

$$x + y \leq 1750$$

$$0 \leq x \leq 1000$$

$$0 \leq y \leq 1500$$

$$(x, y) \in \{0, 225, 450, 675, \ldots\}$$

In which:

x : batches of lager

y : batches of ale

Interior solution!

Why integrality?

Consider the manufacture of television sets. A linear programming model might give a production plan of 205.7 sets per week. In such a model, most people would have no trouble stating that production should be 205 sets per week (or even "roughly 200 sets per week"). On the other hand, suppose we were buying warehouses to store finished goods, where a warehouse comes in a set size. Then a model that suggests we purchase 0.7 warehouse at one location and 0.6 somewhere else would be of little value. Warehouses come in integer quantities, and we would like our model to reflect that fact.

Seemingly Innocuous

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 far beyond restrictions to integral production quantities. With integer variables, one can model logical requirements, fixed costs, sequencing and scheduling requirements, and many other problem aspects.

Formulating an IP is Easy

```
variables
        X(i,j)  Shipment quantities in cases
        Z total Transportation costs in thousands of dollars;

integer variable X;

equations
        cost               define objective function
        supply(i)          observe supply limit at plant i
        demand(j)          satisfy demand at market j ;

cost..          Z =E= sum((i,j), c(i,j)*X(i,j)) ;

supply(i)..     sum(j, X(i,j)) =L= a(i) ;

demand(j)..     sum(i, X(i,j)) =g= b(j) ;

Model transport /all/ ;

X.LO(i,j) = 0;

solve transport using mip minimizing Z ;
```

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 surprisingly often. Such problems are called pure (mixed) 0-1 or *binary* programming problems.

Outline

- Convex Programs
- Discrete and nonconvex models
- Mixed-integer programming
- A trivial example
- A few less trivial examples

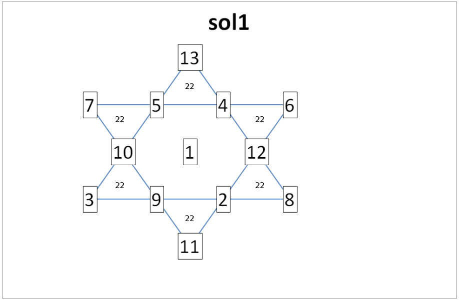
Board Game

The Christmas board game "22"[1] involves a board with 13 holes and 13 pegs which fit in the holes. The pegs are numbered from 1 to 13. Hole are situated at the 12 intersection points on a six-pointed star and in the center of the star. To play the game, a peg is inserted in each hole.

[1] The game is currently on sale in the Christmas market at the Zürich train station.

Winning 22

A winning configuration is one in which the sum of values for each of the six triangles sums to 22. Here, for example, is a winning assignment:



GAMS Code

```
$title  Board Game "22" -- Slow Version Ignores Symmetry

set     n0      All nodes (include center node 0) /n0*n12/,
        n(n0)   Assigned nodes /n1*n12/
        p       Pieces /1*13/,
        t       Triangles /t1*t6/

        tri(t,n)        Assignment of triangles to nodes
                 /      t1.(n1,n3,n4),
                        t2.(n4,n5,n7),
                        t3.(n7,n10,n11),
                        t4.(n9,n10,n12),
                        t5.(n6,n8,n9),
                        t6.(n2,n3,n6)/,
        s               All possible solutions /sol1*sol5000/,
        ss(s)           Solutions which have been found,
        cut(s,p,n)      Cut renders prior solutions infeasible;

cut(s,p,n) = no;
ss(s) = no;
```

Declarative GAMS Code: The Model

```
variable        OBJ     Vacuous objective;

binary
variable        Z(p,n)  Assignment of piece p to node n;

equations       objdef, used, filled, linesum, othersol;

objdef..        OBJ =e= 0;

used(p)..       sum(n, Z(p,n)) =L= 1;

filled(n)..     sum(p, Z(p,n)) =E= 1;

linesum(t)..    sum((p,tri(t,n)), p.val*Z(p,n)) =e= 22;

othersol(ss)..  sum(cut(ss,p,n), Z(p,n)) =l= 11;

model ip /all/;

solve ip using mip maximizing obj;
```

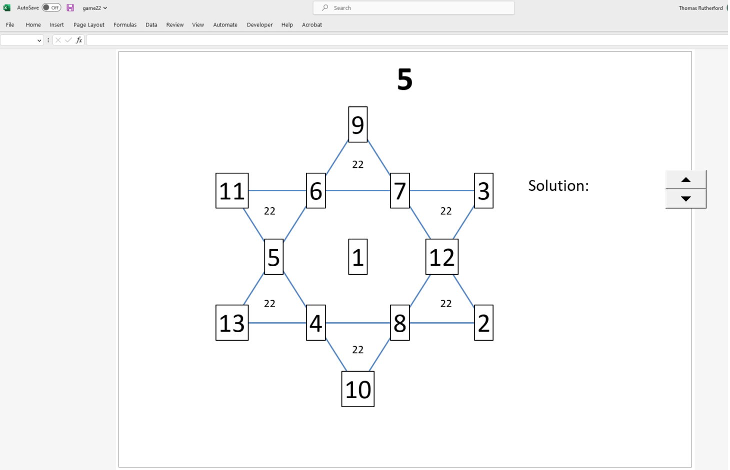
Procedural GAMS Code: Find All Solutions

```
parameter       nsol            Solution count /0/,
                sollist(p,*,*)  Solution assignments;

*       Restrict model output to improve performance:

ip.limrow=0; ip.limcol=0; option solprint=off;

*       We need to solve a number of models -- to make this go
*       more rapidly, let GAMS load the solver in memory to
*       avoid the cost of reloading the library each time:

ip.solvelink = %solvelink.LoadLibrary%;

*       Outer loop over s=sol1 provides an "anchor point" for
*       assignments to the offet reference s+nsol:

loop(s$sameas(s,"sol1"),

*       Find solutions with peg pp in the center hole:

  loop(pp,

*       Omit pp from the assignment into nodes other than 0:

        Z.FX(pp,n) = 0;

*       First solution:

        solve ip using mip maximizing obj;
```

Procedural GAMS Code: Find All Solutions

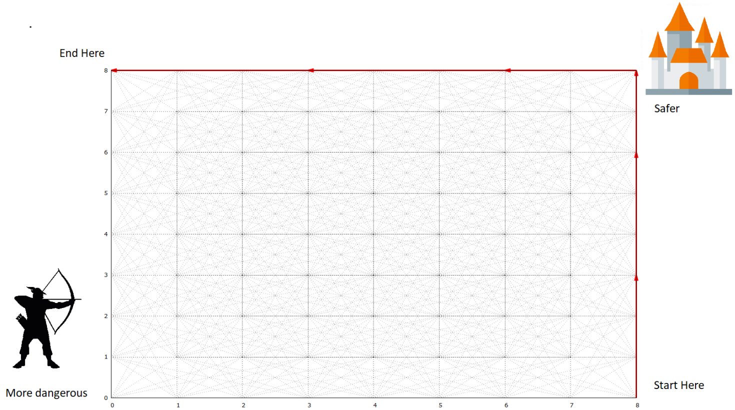
```
*       While we have a solution and we have found 9 or
*       fewer solution with peg pp in the center, continue
*       solving:

        while (((ip.modelstat<>10) and (nsol<=9)),


*       We have a solution in hand at this.  Add this to the
*       list of cuts, using "s+nsol" to refer to this new solution:

          cut(s+nsol,p,n)$Z.L(p,n) = yes;

*       Add the new solution to the list of cuts:

          ss(s+nsol) = yes;

*       Keep record of which pegs are in which holes for this
*       solution:

          sollist(pp,"n0",s+nsol) = pp.val;
          loop((p,n)$Z.L(p,n), sollist(pp,n,s+nsol) = p.val; );

*       See if we can find another solution:

          nsol = nsol + 1;
          solve ip using mip maximizing obj;

        );
```

Visualization of Results



Route Choice with Risk

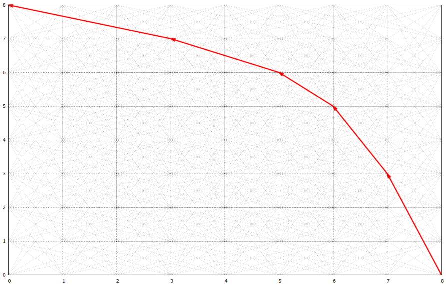
Last lecture we formulated a linear programming model with dual objectives: minimize distance and minimize risk. I mentioned at the end of class that there is a compact nonlinear programming formulation which is efficient but non-convex. Convexity is a typical in economic equilibrium models but it is often elusive in general decision problems.

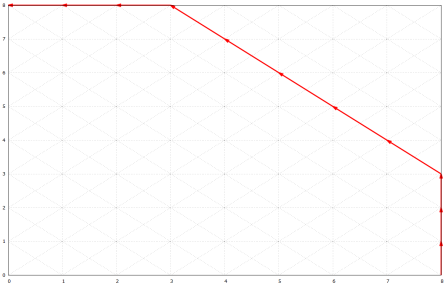


Maximum Weight Flow LP Model in GAMS

```
variable        OBJ             Objective function;

nonnegative
variables       X               Route choice;

equations       conservation, objdef;

objdef..
        OBJ =e= sum(a, X(a) * dist(a)*(lamda - (1-lamda)*log(p(a))));

conservation(k)..

        sum(a(i,k), X(a)) + 1$start(k) =e= sum(a(k,j),X(a)) + 1$end(k);

model routechoice /conservation, objdef/;
```

In the LP Model, Discretization Matters ($\lambda = 0.4$)





GAMS Code: The NLP Formulation

```
set                pt      Points on the route /0*20/,


variable        OBJ                 Objective function;

nonnegative
variables       X(pt)               Route choice -- X dimension
                Y(pt)               Route choice -- Y dimension
                P(pt,h)             Proximity from hazard h
                R(pt,h)             Risk from hazard h on passage from pt-1 to pt
                Z                   Overal hazard rate
                L                   Overall length of route,
                D                   Distance between points;

equations       objdef, length, proximity, risk, hazard, distance, monotone;

*       Define the objective as a weighted sum of travel distance and probability
*       of reading the target:

objdef..        OBJ =e= lamda * L + (1-lamda) * Z;
```

GAMS Code: The Model (cont.)

```
*       Total length of the path:

length..                L =e= sum(pt, sqrt(sqr(X(pt+1)-X(pt))+
                                   sqr(Y(pt+1)-Y(pt)))) - 1;

*       Hazard of the path = 1 - probability (no failure):

hazard..                Z =e= 1 - prod(pt, prod(h,1-R(pt,h)));

*       Distance between point pt and pt+1:

distance(pt+1)..        sqr(D) =e= sqr(X(pt+1)-X(pt)) + sqr(Y(pt+1)-Y(pt));
```

GAMS Code: The Model (cont.)

```
*       Proximity of point pt to hazard h:

proximity(pt+1,h)..

        P(pt+1,h) =e= sqrt(sqr((X(pt+1)+X(pt))/2-hloc(h,"x")) +
                           sqr((Y(pt+1)+Y(pt))/2-hloc(h,"y")));

*       Risk imposed on passage from point pt to pt+1:

risk(pt+1,h)..          R(pt+1,h) =e= exp(-0.5*sqr(P(pt+1,h)/var(h)))*D;

model routechoice /objdef, length, proximity, risk, hazard, distance/;
```

GAMS Code: Bounds and Initialization

```
*       No risk on the arc leading to point 0:

R.FX("0",h) = 0;

*       Upper bound on total length:

L.UP = 2.2;

*       Points much lie in the unit square:

X.LO(pt) = 0;           X.UP(pt) = 1;
Y.LO(pt) = 0;           Y.UP(pt) = 1;

D.LO     = 0.001;       D.UP = 1;

*       Assign some initial values:

P.L(pt,h) = 0.1;  D.L = 1/20; X.L(pt) = 1/2;    X.L(pt) = 1/2;

*       Start in the southeast (1,0) and traverse to the northwest (0,1):

X.FX("0")  = 1;         Y.FX("0")  = 0;
X.FX("20") = 0;         Y.FX("20") = 1;
```

GAMS Code: Loop over Weights

```
set     scn              Scenarios to compare /0,30,60,90,minRisk/,
        theta(scn)       Angle which determines lamda (degrees) /0,30,60,90/

parameter       rc(scn,pt,xy)   Route choice,
                step(scn)       Step length;

loop(theta,

        lamda = cos(pi*theta.val/180)/(cos(pi*theta.val/180)+sin(pi*theta.val/180));

        solve routechoice using nlp minimizing obj;

*       Report the location of each step on this route:

        step(theta) = D.L;
        rc(theta,pt,"x") = X.L(pt) + eps;
        rc(theta,pt,"y") = Y.L(pt) + eps;
);
```

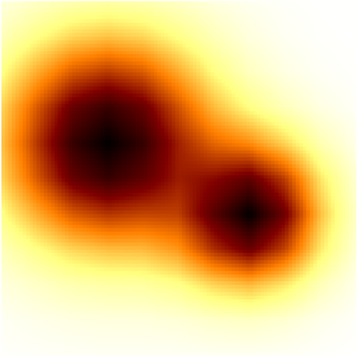
GAMS Code: Finding the Global Optimum

```
*       Find a route which runs through the northeast corner:

Y.FX("10") = 1; X.FX("10") = 1;

*       Almost all weight on distance:

lamda = 0.01;
solve routechoice using nlp minimizing obj;

*       Verify local optimality.

D.LO = 0; D.UP = 1;
Y.UP("10") = 1; X.UP("10") = 1;

solve routechoice using nlp minimizing obj;

rc("minRisk",pt,"x") = X.L(pt) + eps;
rc("minRisk",pt,"y") = Y.L(pt) + eps;
step("minRisk") = D.L;
```

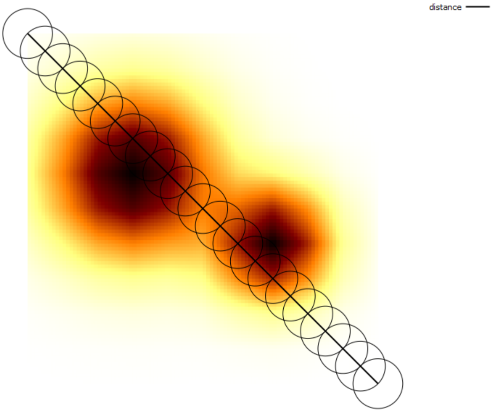
An NLP Formulation

```
set     xy      Planar dimensions /x,y/;

parameter       lamda   Weight on speed (1=shortest path) /1/;

set     h               Hazards /h1,h2/;

table   hloc(h,xy)      Hazard locations

        x       y
h1      0.3     0.6
h2      0.7     0.4;

parameter       var(h)  Variance /h1 0.15, h2 0.1/;
```

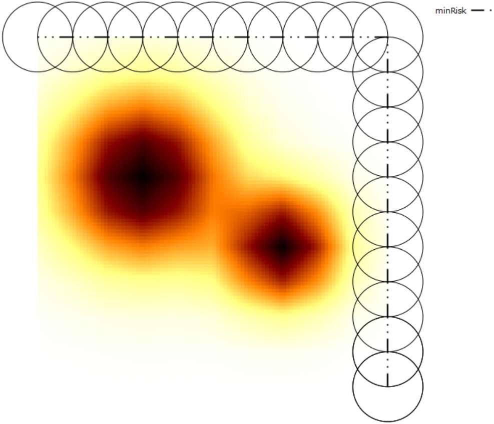
Risk Geometry



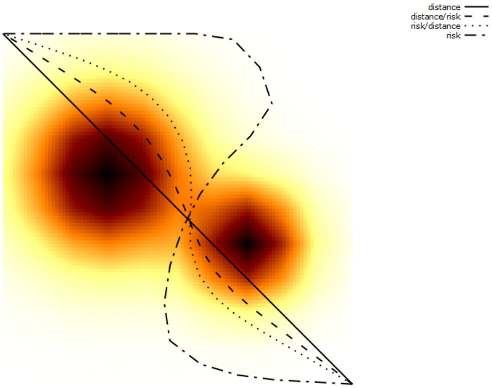
The Shortest Path



The Safest Path

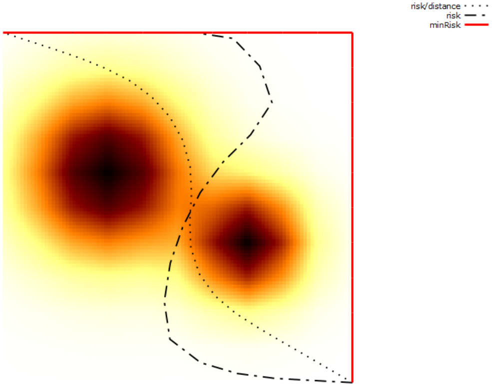


A Sequence of Solutions for $\theta = \{0, 30, 60, 90^\circ\}$



Local versus Global

The starting point determines which solution is returned. When initiating from the $\theta = 60^\circ$ solution, we find a local optimum. If we initiate from a boundary route, we find the global solution:



Cash Flow Models

Companies a liability stream reaching several years into the future:

- payouts on insurance products
- home owner mortgages (with fxed or variable payments)
- lottery payouts

For this reason it can be useful to determine a portfolio of bonds (obligations) whose cash-fows replicate that of the liability stream.

One way for insurance companies to demonstrate solvencyis to assess the fair market value of their liabilities. This can be done by replicating portfolio consisting of default-free bonds, such as Treasuries.

The Deterministic Case

$T = \{0, \ldots, m\}$ The set of time periods, for instance measured in years, from $t = 0$ (now) to $t = m$ (the time horizon).

$U = \{1, \ldots, n\}$ The *universe* of assets under consideration for inclusion in the portfolio.

$F_{i,t}$ The cash flow arising from asset i at time t.

L_t The liability due in period $t \geq 1$.

Cash Flows

The cash flows $F_{i,t}$ can be both positive and negative; by convention, a positive number indicates incoming cash and a negative number outgoing cash.

For a *bond* which is purchased today at price P_i per unit *face value*, with an annual *coupon payment* of c_i and a *maturity* of 4 years we would have

$$F_{i,\cdot} = (-P_i, c_i, c_i, c_i, 1 + c_i, 0, \ldots, 0).$$

Cash Flow Matching Model

Decision variables for the fund manager are $x_i \quad i \in U$, the amount in face value of bond i to be purchased:

$$\min \lambda$$

subject to:

$$\sum_i F_{i,0} x_i + \lambda \geq 0$$

$$\sum_i F_{i,t} x_i \geq L_t \quad \forall t$$

$$x_i \geq 0$$

Reinvesting and Borrowing

Short-term reinvesting and borrowing can be modeled by adding variables r_t and b_t to represent the amount of cash reinvested or borrowed from period t to $t+1$:

$$\min \lambda$$

subject to:

$$\sum_i F_{i,0} x_i + b_0 + \lambda \geq r_0$$

$$\sum_i F_{i,t} x_i + (1+\rho_t) r_{t-1} + b_t \geq L_t + r_t + (1+\beta_t) b_{t-1} \quad t \geq 1$$

$$b_m = 0$$

$$x_i, r_t, b_t \geq 0$$

Tradeability Considerations

Large institutional bond investors are usually interested in purchasing large blocks of individual bonds, in round numbers (even lots) of face value. This is primarily because such blocks are more easily traded than smaller or *odd-lot* holdings, but also for liquidity reasons, to avoid the risk of getting stuck with a small holding of a bond with poor liquidity.

We may therefore wish to modify our model to enforce:

1. even-lot purchases, i.e., face-value amounts in multiples of, say, $100,000, and/or
2. minimum-lot purchases, i.e., purchase either none or at least some amount, say $500,000, of each bond.

Suppose decision variables x_i denote the number of \$100 bonds to purchase and lots in multiples of \$100,000 to be implemented. Add *integer variables* y_i and constraints

$$x_i = (100000/100)y_i$$

to the cash flow model.

Minimum-lot purchases can be implemented using *binary variables*, $b_i \in \{0, 1\}$ and

$$(500000/100)b_i \leq x_i$$

and

$$x_i \leq M \; b_i$$

where M is a large number (*big M*). These constraints assure that when $b_i = 0$ then $x_i = 0$, and $b_i = 1$ then $x_i \geq (500000/100)$.

Transactions Costs

Investments may include transactions costs such as brokerage fees and bid-ask spreads. Portfolios containing small amounts of a large number of securities introduce administrative costs. We distinguish between two types of transactions costs: *fixed* and *variable*. Fixed costs are introduced with binary variables $z_i \in \{0, 1\}$ and *big M* constraints:

$$\text{Fixed cost} = F \sum_i z_i$$

$$x_i \leq M\ z_i$$

Variable, or proportional, transactions costs, on the other hand, are easily modeled without any penalty in model complexity.

Even Lot Purchases: Mixed Integer Programming

Suppose decision variables x_i denote the number of \$100 bonds to purchase and lots in multiples of \$100,000 to be implemented. Add *integer variables* y_i and constraints

$$x_i = (100000/100)y_i$$

to the cash flow model.
Minimum-lot purchases can be implemented using *binary variables*, $b_i \in \{0, 1\}$ and

$$(500000/100)b_i \leq x_i$$

and

$$x_i \leq M\ b_i$$

where M is a large number (*big M*). These constraints assure that when $b_i = 0$ then $x_i = 0$, and $b_i = 1$ then $x_i \geq (500000/100)$.

Transactions Costs

Investments may include transactions costs such as brokerage fees and bid-ask spreads. Portfolios containing small amounts of a large number of securities introduce administrative costs. We distinguish between two types of transactions costs: *fixed* and *variable*. Fixed costs are introduced with binary variables $z_i \in \{0, 1\}$ and *big M* constraints:

$$\text{Fixed cost} = F \sum_i z_i$$

$$x_i \leq M \; z_i$$

Variable, or proportional, transactions costs, on the other hand, are easily modeled without any penalty in model complexity.