

# Stochastic Programming Project

Konrad Borys

## Model for Optical Fiber Manufacturing

### 1. Introduction

Optical fibers are made of solid rods of glass called preforms. The ends of the preforms are heated and fibers are drawn from them. The fibers may randomly break during the process. The random lengths of fibers produced are cut into standard lengths immediately after their production.

Time is divided into periods. Demand for each product is assumed to be known for each period and deliveries are made at the ends of periods. We assume that a large number of preforms are to be processed during each production period. In the present model we consider only one production period.

Our goal is to find such cutting rule that minimizes the expected loss due to unsatisfied demand.

### 2. Random Yield

The collection of semi-finished products produced from a given number of preforms will be called yield. We assume that locations of microscopic defects in the fiber that cause a breakage form a homogenous Poisson process.

We characterize a yield of a single preform by the number of semi-finished products of different lengths.

Let

$H$  – total length of the fiber that can be drawn from a single preform  
 $\varphi_h$  – number of those semi-finished products which lengths fall into interval  $[h, h+1)$ , where  $1 \leq h \leq H$

Then the random yield of a single preform is characterized by

$$\varphi = [\varphi_1, \dots, \varphi_H]^T$$

And the random yield of  $N$  preforms is

$$\Phi = \varphi^{(1)} + \dots + \varphi^{(N)}$$

### 3. Cutting Rule

Let  $h_1 < \dots < h_m$  be the standard product lengths and  $h_1, \dots, h_m$  are integers. Let  $h$  be a length of a semi-finished product,  $1 \leq h \leq H$ .

A cutting pattern for the length  $h$  is a vector

$$v = [v_1, \dots, v_m]^T$$

such that

$$\sum_{k=1}^m v_k h_k \leq h, \text{ and } v_1, \dots, v_m \geq 0, \text{ integer}$$

The cutting pattern means that we cut  $v_k$  pieces of a standard length  $h_k$ .

Convex combinations of cutting patterns for length  $h$  are called generalized cutting patterns. We define a cutting rule  $A$  as  $m$  by  $H$  matrix, where a column  $h$  is generalized cutting pattern for length  $h$ .

Suppose the yield  $\phi$  is cut according to  $A$ . Then the number of products of length  $h_k$  will be approximately  $a^k \phi$ , where  $a^k$  is  $k^{\text{th}}$  row of  $A$ .

### 4. Formulation of the Cutting Rule Finding Problem

Let  $b = [b_1, \dots, b_m]^T$  be a demand for products of length  $h_1, \dots, h_m$ .  
For a given cutting rule  $A$  and yield  $\phi$

$$[b - A\phi]_+$$

is the unsatisfied demand.

Let  $q = [q_1, \dots, q_m]^T$  be a cost vector, then the loss due to unsatisfied demand is

$$q^T [b - A\phi]_+$$

Our aim is to choose a cutting rule that minimizes the expected loss

$$\begin{aligned} & \text{minimize } E[q^T [b - A\phi]_+] \\ & \text{subject to } A \text{ is a cutting rule} \end{aligned}$$

## 5. Computation of the Value of the Objective Function

By a multidimensional limit theorem  $\phi$  has H-variate normal distribution with an expectation  $\mu$  and a covariance matrix C.

$$\Delta(A) = E[q^T [b - A\phi]_+] = \sum_{k=1}^m q_k (b_k - a^k \mu) \Phi\left(\frac{b_k - a^k \mu}{\sqrt{a^k C a^k}}\right) + q_k \frac{\sqrt{a^k C a^k}}{\sqrt{2\pi}} e^{-\frac{1}{2a^k C a^k} (b_k - a^k \mu)^2}$$

## 6. Computation of the Gradient of the Objective Function

$\Gamma(A)$  is m by H matrix, and an entry in k<sup>th</sup> row and h<sup>th</sup> column is

$$q_k v_1 \Phi\left(\frac{b_k - v_2}{\sigma_2}\right) + q_k \frac{\rho \sigma_1}{\sqrt{2\pi}} e^{-\frac{1}{2\sigma_2^2} (b_k - v_2)^2}$$

where

$$\begin{aligned} v_1 &= e_h^T \mu \\ v_2 &= (a^k)^T \mu \\ \sigma_1^2 &= e_h^T C e_h \\ \sigma_2^2 &= (a^k)^T C a^k \\ \rho &= \frac{(a^k)^T C e_h}{\sigma_1 \sigma_2} \end{aligned}$$

## 7. Algorithm – Feasible Direction Method

0. Initial cutting rule  $A = 0$
1. Compute  $\Gamma(A)$
2. For each column ( $h=1..H$ ) of  $\Gamma(A)$  solve an integer knapsack

$$\begin{aligned} & \text{minimize } \Gamma_h(A) d^h \\ & \text{subject to} \\ & \sum_{k=1}^m d_k^h h_k \leq h \\ & d_1^h, \dots, d_m^h \geq 0, \text{ integer} \end{aligned}$$

3. Let  $D = [d^1, \dots, d^H]$   
Then  

$$D' = D - A$$
 Is a feasible direction that minimizes the directional derivative  $\Gamma(A)$
4. Solve a one-dimensional convex problem

$$\begin{aligned} & \text{minimize } \Delta(A + \lambda D') \\ & \text{subject to} \\ & 1 \leq \lambda \leq 1, \text{ real} \end{aligned}$$

5. If the length of the step  $\lambda$  is very small  
 return  $A$  near-optimal cutting rule  
 else  

$$A = A + \lambda D'$$
  
 go to 1

## 8. Tests

### Small example1

In this example the parameters are:

$H = 5$ ,  $m = 2$ , probability  $q = 0.97$ ,  $N = 200$

demand  $b = [160.0, 280.0]$

costs  $q = [2.2, 3.4]$

standard lengths:

$h_1 = 2$

$h_2 = 4$

### Solution

# ITERATIONS = 8031

$\lambda = 5.3783e-008$

objective value = 566.6934

$A = \begin{bmatrix} 0 & 1 & 1 & 0.7392 & 0.7392 \\ 0 & 0 & 0 & 0.6304 & 0.6304 \end{bmatrix}$

### Small example2

The following parameters are given in this example:

$H = 5$ ,  $m = 2$ ,  $N = 300$ , probability  $q = 0.97$

The stopping tolerance (epsilon) for lambda is 0.0001

The demand vector (b) is [126.0, 237.0]

The prices (q) are [2.2, 3.4]

The standard fiber lengths (h1, h2) are 2 and 4.

### Solution

# of iterations = 1856

$\lambda = 1.2242e-008$

objective value = 34.8444

$A = \begin{bmatrix} 0 & 1 & 1 & 0.3221 & 0.3221 \\ 0 & 0 & 0 & 0.8389 & 0.8389 \end{bmatrix}$

## 9. Matlab code

### **function [Expectation,Covariance] = step0(H,q,N)**

```
miu = zeros(H,1);
for h=1:H
    miu(h) = q^h - log(q) * (q^h) * (H-h);
end

for h=1:H-1
    Expectation(h) = ( miu(h) - miu(h+1) ) * N;
end
Expectation(H) = miu(H) * N;

miu2 = zeros(H,H);
for h=1:H
    for g=1:H
        if H-h-g > 0
            miu2(h,g) = (log(q))^2 * q^(h+g) * (H-h-g)^2 - 2 * log(q) * q^(g+h) * (H-h-g) + miu(max(h,g));
        else
            miu2(h,g) = miu(max(h,g));
        end
    end
end
end
%-----
ex_mh_mg = zeros(H,H);

ex_mh_mg(H,H)=miu(H);
for h=1:H-1
    for g=1:H-1
        ex_mh_mg(h,g) = miu2(h,g) - miu2(h+1,g) - miu2(h,g+1) + miu2(h+1,g+1);
    end
end
%=====
Covariance =zeros(H,H);
for i=1:H
    for k=1:H
        Covariance(i,k) = N*ex_mh_mg(i,k) - Expectation(i)*Expectation(k)/N ;
    end
end
end
```

## function [gamma] = Gamma(A,q,b,Expectation,Covariance)

```
m = size(A,1);
H = size(A,2);

for k=1:m
    p = A(k,1:H);
    if p == 0
        for h=1:H
            gamma(k,h) = -q(k) * Expectation(h);
        end
    else
        miu2 = p*Expectation';
        sigma2_square = p*Covariance*p';
        sigma2 = sqrt(sigma2_square);
        for h=1:H
            miu1 = Expectation(h);
            sigma1 = sqrt( Covariance(h,h) );
            ro = p*Covariance(1:H,h);
            ro = ro / (sigma2 * sigma1) ;

            gamma(k,h)=-q(k)*miu1*normcdf( (b(k)-miu2)/sigma2 ) + q(k)*ro*sigma1*(1/sqrt(2*pi))*exp( -
1/(2*sigma2_square))*(b(k)-miu2)^2 );
        end
    end
end
end
```

## function [D] = step1h(A,Gamma,hm)

```
M = 1000;
m = size(Gamma,1);
H = size(Gamma,2);
D = zeros(m,H);
```

```
D(1,2)=1;
D(1,3)=1;
t1 = [ 1 0];
t2 = [ 2 0];
t3 = [ 0 1 ];
```

```
for h=4:5
    obj = Gamma(:,h);
    if t1*hm<=h
        max = t1 * obj;
        x = t1;
    end
    if t2*hm<=h
        o2 = t2 * obj;
        if o2 < max
            max = o2;
            x = t2;
        end
    end
    if t3*hm<=h
        o3 = t3 * obj;
        if o3 < max
            max = o3;
            x = t3;
        end
    end
    D(:,h) = x;
end
D;
D = D - A;
```

## function f = Delta(x)

```
load('DATA', 'A', 'DD', 'q', 'b', 'Ex', 'Cov')
```

```
m = size(A,1);
f=0;
for k=1:m
    a = A(k,:);
    d = DD(k,:);
    if (a==0 & d==0 )
        f=f+b(k)*q(k);
    else
        miu = (a + x * d)*Ex';
        sigma_suare = (a + x *d )*Cov*(a + x *d );
        FF = normcdf( (b(k)- miu)/sqrt(sigma_suare) ) ;
        expon = exp( (-1/(2*sigma_suare)) * (b(k) - miu )^2 );

        f = f + q(k)*( b(k)- miu ) * FF + q(k)* sqrt(sigma_suare)/sqrt(2*pi) * expon;
    end
end
f;
```

## **function [success] = main(q,b,hm,Ex,Cov,iterations)**

```
load('DATA_A', 'A')

for i=1:iterations
    Gamma = Gamma(A,q,b,Ex,Cov);
    DD = step1h(A,Gamma,hm);

    %step2
    save('DATA', 'A', 'DD', 'q', 'b', 'Ex', 'Cov');
    x = fmincon(@Delta,0.1,[],[],[],[],0,1);

    if x<0.0001
        i
        x
        Delta(x)
        A
        A = A + x*DD;
        return
    end
    A;
    A = A + x*DD;
end
i
A
x
Delta(x)
save('DATA_A', 'A')
```

## **function small\_example1()**

```
H = 5;
qq = 0.97;
N = 300;
[Ex,Cov]=step0(H,qq,N);

hm = [ 2; 4 ];
b = [ 126 237 ];
q = [ 2.2 3.4 ];

A= [0, 0, 0, 0, 0;
     0, 0, 0, 0, 0]
save('DATA_A', 'A');
main(q,b,hm,Ex,Cov,3000);
```