Revenue Management

Tool : Optimization (Prescriptive analytics)

The Analytics Edge

In the late 1970's, deregulation in the airlines industry, provided opportunities but challenges for many of the airline carriers to increase profit and stay ahead of the competition. Charter airlines started selling much cheaper tickets than airlines such as American Airlines. Managing the seats sold effectively was realized to be important & revenue management techniques were used to increase revenues. Today revenue management & the use of analytics for this purpose is affecting many industries including hotels, online retailers. We will use the example of American Airlines & how it used techniques of revenue management to generate revenue. We will also discuss extension of these ideas to pricing decisions in revenue management. The key tool that is used is the application of optimization techniques for prescriptive analytics.

Revenue management increases company profits by using analytics to ~~...~~ try and sell the right product to the right customer at the right time for the right price. After deregulation in the airlines industry in the U.S, companies had the flexibility to explore different pricing & routing options & the best mix of passengers to maximize revenue. Prior to deregulation, airlines could only fly certain routes & fares were determined by the federal Civil Aeronautics Board based on mileage & costs.

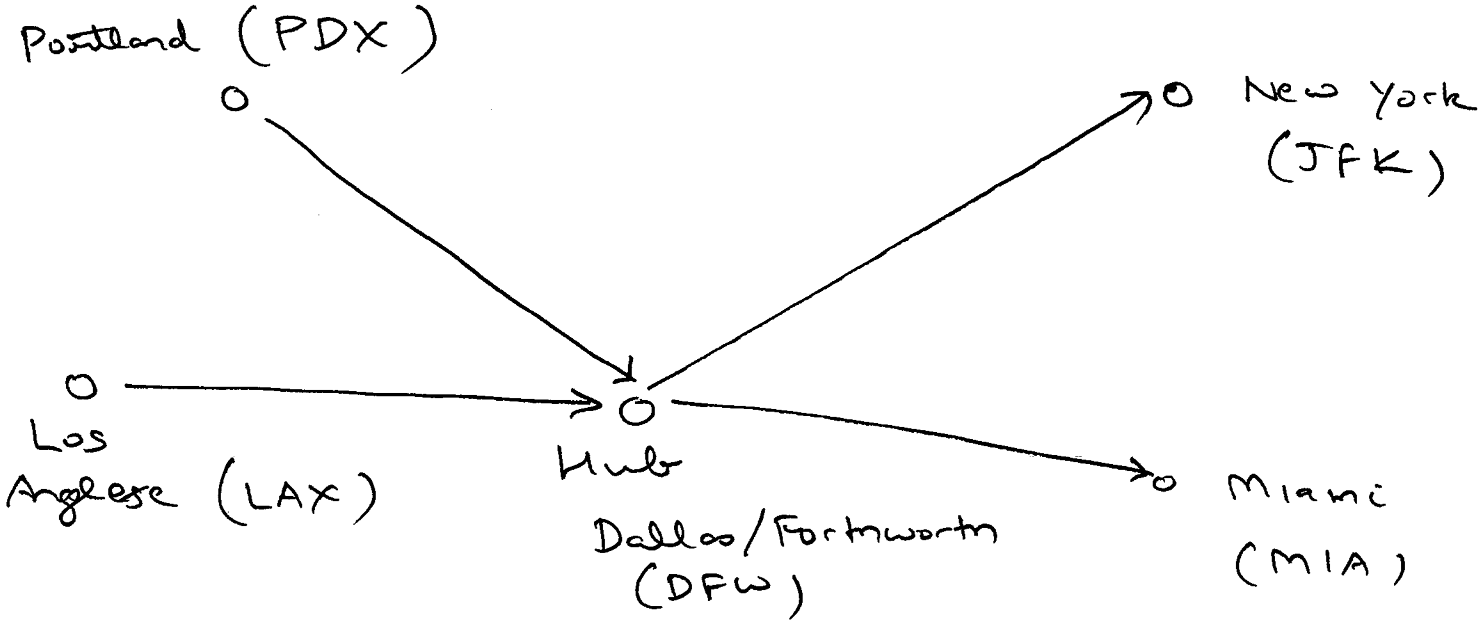
Among some of the features of the airlines industry that makes it suitable for revenue management are: 1) fixed capacity (seats), 2) high fixed costs but low variable costs, 3) variety of customer types (price sensitive, luxury travelers) and 4) ability to sell tickets to customers without seeing what other customers paid.

American Airlines is believed to be one of the first airlines that successfully implemented a revenue management system.

The scale of this problem can be huge since reservations are made often months in advance to a day before, multiple departures for an airline carrier and hence the system needs to be computerized just to handle the volume of business.

We discuss some of these challenges taken from an influential paper "Yield Management at American Airlines", Interfaces, 1992. One of the problems discussed there is the <u>discount allocation</u> problem which is the problem of determining the number of discount fares to offer on the flight.

The tradeoff is to offer discounts to try & fill seats or else the flight will fly with empty seats (fixed cost) but limit it so that you can fill seats with possibly late arriving higher revenue passengers.



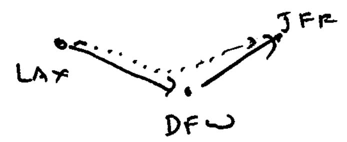
One hub, four spoke network where passengers travel from PDX & LAX to DFW directly or connect at DFW to travel to JFK & MIA. ~~to~~ Thus there are four types of passenger routes in this network. Furthermore there are two classes of passengers, one who book in advance & want to pay less & a second class who book late but are willing to pay a lot more. This inspired the airlines to offer two products: 1) more expensive ticket that can be purchased any time, no restriction & fully refundable, 2) cheaper ticket, to be bought atleast 3 weeks in advance, penalties for changing ..

We consider a simple version of the problem first before discussing extensions.

Assume the marketing department has already decided the prices & we are interested in deciding how many seats to reserve in each fare class & itinerary. For simplicity, we consider only 3 flights: LAX → DFW, LAX → JFK, DFW → JFK. Only two legs are used here in terms of flights. For simplicity consider the data:

Capacity LAX – DFW flight 300

Capacity DFW – JFX flight 200



		Revenue \$	Demand	Variable
LAX → DFW	Regular	100	20	x_1
	Discount	90	40	x_2
	Saver	80	60	x_3
LAX → JFK	Regular	215	80	y_1
	Discount	185	60	y_2
	Saver	145	70	y_3
DFW → JFK	Regular	140	20	z_1
	Discount	120	20	z_2
	Saver	100	30	z_3

Linear program

$$\max \quad 100x_1 + 90x_2 + 80x_3 + 215y_1 + 185y_2 + 145y_3 + 140z_1 + 120z_2 + 100z_3$$

s.t.

$$x_1 + x_2 + x_3 + y_1 + y_2 + y_3 \le 300$$

$$y_1 + y_2 + y_3 + z_1 + z_2 + z_3 \le 200$$

$$0 \le x_1 \le 20$$

$$0 \le x_2 \le 40$$

$$0 \le x_3 \le 60$$

$$0 \le y_1 \le 80$$

$$0 \le y_2 \le 60$$

$$0 \le y_3 \le 70$$

$$0 \le z_1 \le 20$$

$$0 \le z_2 \le 20$$

$$0 \le z_3 \le 30$$

Optimal solution is $x_1 = 20$, $x_2 = 40$, $x_3 = 60$, $y_1 = 80$, $y_2 = 60$, $y_3 = 40$, $z_1 = 20$, $z_2 = 0$, $z_3 = 0$.

Total revenue = \$47,300

What would a greedy method do?

Give priority to highest revenue customers & greedily allocate them.

Here we would set $\hat{y}_1 = 80$, $\hat{y}_2 = 60$ & $\hat{y}_3 = 60$ (we use only 60 from supersaver since capacity of flight from DFW to JFK is only 200). We still have 100 seats on LAX to DFW flight & would choose $\hat{x}_1 = 20$, $\hat{x}_2 = 40$, $\hat{x}_3 = 40$. The total revenue in this case is \$ 46,000 which is about 3% lesser total revenue than optimum.

Aggregating over all flights, all fare classes, this loss of revenue could be significant.

Practical implementation

There are some potential challenges with this model.

1) As soon as regular priced tickets are sold, only discount tickets are available though it is possible that some customers might still be willing to buy at a regular price. Airlines use "nesting control" where seats reserved for discount classes are made available to more expensive classes. Namely make all tickets available to regular customers, discount & super saver tickets to discount customers & only super saver tickets are available to super saver customers.

2) An advantage of a LP model is that it provides a shadow price for each constraint (dual variable). This reflects the network revenue value of last seat on each leg of the flight for the capacity constraint. These prices also referred to as bid prices are used in practice to decide whether or not an additional seat should be offered for a particular flight & fare class.

For example if leg 1 dual variable = P_1 &

leg 2 dual variable = P_2 then for any new

fare request on the leg 1 + leg 2 itinerary if

fare willing to pay $\geq P_1 + P_2$, it is accepted,

else reject.

3) The data is often changing with time & dynamic

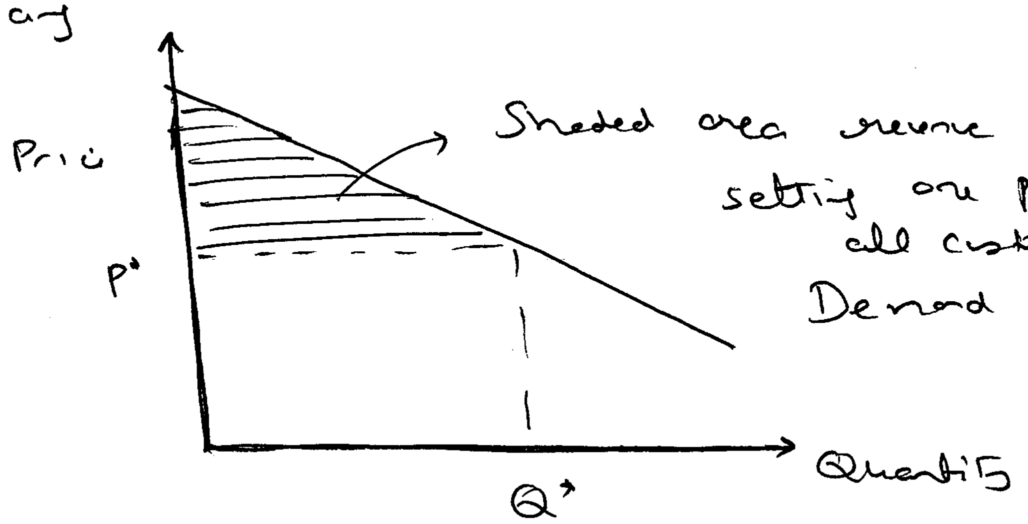
models with randomness in demand needs to

be incorporated in such an implementation.

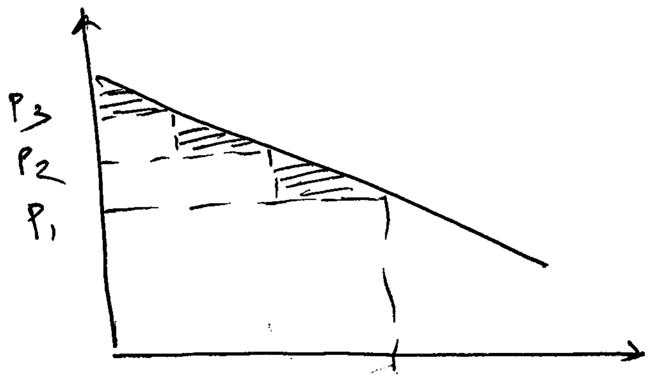
```
Pkg.add("JuMP")

Pkg.add("Gurobi")

using JuMP, Gurobi

m = Model(solver=GurobiSolver())

@variable(m,x[1:3]>=0)

@variable(m,y[1:3]>=0)

@variable(m,z[1:3]>=0)

@constraint(m,constraint1,x[1]+x[2]+x[3]+y[1]+y[2]+y[3] <=
300)

@constraint(m,constraint2,y[1]+y[2]+y[3]+z[1]+z[2]+z[3] <=
200)

@constraint(m,x[1]<=20)

@constraint(m,x[2]<=40)

@constraint(m,x[3]<=60)

@constraint(m,y[1]<=80)

@constraint(m,y[2]<=60)

@constraint(m,y[3]<=70)

@constraint(m,z[1]<=20)

@constraint(m,z[2]<=20)

@constraint(m,z[3]<=30)

@objective(m,Max,100*x[1]+90*x[2]+80*x[3]+215*y[1]+185*y[2]+
145*y[3]+140*z[1]+120*z[2]+100*z[3])

print(m)

solve(m)

getobjectivevalue(m)

getvalue(x)

getvalue(y)

getvalue(z)

getdual(constraint1)

getdual(constraint2)
```

Prescribing the right prices

One of the key aspects of revenue management is pricing



Consider the demand curve for a single product with the price on the y-axis



Suppose we can set three prices for the same product. Then we can reduce the amount of revenue lost by setting a single price

Customers willing to pay more than P_1 but less than P_2 pay P_1 & customers willing to pay more than P_2 but less than P_3, pay P_2.

Pricing products with MNL

Discrete choice models are also used to determine prices of products since it helps describe consumer demand

Consider a set of products denoted as $\{1, \dots, n\}$ and an outside option denoted by 0. The goal of the decision-maker is to price products $1, \dots, n$. These products are substitutes and this is accounted for by choice models

$$\max_{p_1 \geq 0, \dots, p_n \geq 0} \sum_{i=1}^{n} (\underbrace{p_i}_{\text{Price}} - \underbrace{c_i}_{\text{Cost}}) \underbrace{\mathbb{P}\begin{pmatrix}\text{Choosing product} \\ i \text{ given prices} \\ \bar{p}\end{pmatrix}}_{\text{Obtained from logit model}}$$

$$\mathbb{P}\begin{pmatrix}\text{choosing product} \\ i \text{ given prices } \bar{p}\end{pmatrix} = \frac{e^{v_i - p_i}}{\sum_{j=0}^{n} e^{v_j - p_j}}$$

$$\therefore \max_{p_1, \dots, p_n \geq 0} \sum_{i=1}^{n} \frac{(p_i - c_i)\, e^{v_i - p_i}}{\sum_{j=0}^{n} e^{v_j - p_j}} \quad \text{where } v_0 = p_0 = 0$$

This is not an easy objective function to deal with

Alternatively solve it in choice probability variables

$$q_i = \frac{e^{v_i - p_i}}{\sum_{j=0}^{n} e^{v_j - p_j}} \qquad \forall j = 1, \dots, n$$

$$q_0 = \frac{1}{\sum_{j=0}^{n} e^{v_j - p_j}} \qquad \text{where } v_0 = p_0 = 0$$

Then $\quad v_i - p_i = \log(q_i) - \log(q_0)$

$$\therefore \quad p_i = v_i - \log(q_i) + \log(q_0)$$

$$\max \sum_{i=1}^{n} \left(v_i - \log(q_i) + \log(q_0) - c_i\right) q_i$$

$\sum_{i=0}^{n} q_i = 1$

$q_i \geq 0 \;\; \forall i = 0, \dots, n$

Note this simplifies to:

$$\max \sum_{i=1}^{n} \left(v_i - c_i - \log(q_i)\right) q_i + (1-q_0)\log(q_0)$$

$\sum_{i=0}^{n} q_i = 1$

$q_i \geq 0 \;\; \forall i = 0 \dots n$

Note $-q \log(q)$ is concave in q variable.

Also $(1-q_0)\log(q_0)$ is concave in q_0 variable

Hence this problem is efficiently solvable using convex optimization

$$\frac{d}{dq}\left(-q \log q\right) = -1 - \log q$$

$$\frac{d^2}{dq^2}\left(-q \log q\right) = -\frac{1}{q} \leq 0$$

$$\frac{d}{dq}\left((1-q) \log q\right) = \frac{1-q}{q} - \log q$$

$$\frac{d^2}{dq^2}\left((1-q) \log q\right) = \frac{q(-1) - (1-q)}{q^2} - \frac{1}{q}$$

$$= \frac{-1}{q^2} - \frac{1}{q} \leq 0$$

In practice, how might companies choose the v?

For example, online retailers can easily change the prices & observe the market share for products. Thus they change prices & observe market shares. By doing this many times they can try & estimate v

Prices			Market share	
1	n		1	n
p_1	p_n	$\longrightarrow$	x_1	x_n
q_1	q_n		y_1	y_n
r_1	r_n		z_1	z_n

We can then do simple regression to estimate the valuation of ~~prices~~ products.

Also in some cases, companies such as GM use complex simulators to predict market shares. How to optimize prices is a problem to solve?

```
n=20

valuation =[46.7
46.5
46.9
43.8
48.8
42.7
46.4
39.8
37.6
33.5
27.5
35.4
33.5
37.6
40.8
35.2
38.7
42.3
36.8
41.8
]

baseprice = [45.5
46
45
43
47
42
45
39
37
33
27
35
33
37
39
35
38
41
36
40
]

cost = [34
35
33
32
35
31
34
35
34
29
24
```

```
32
30
34
33
30
32
35
30
33
]

using JuMP, Ipopt

m = Model(solver=IpoptSolver())

@variable(m,p[1:n]>=0)

@variable(m,normp>=0)

@NLobjective(m,Max,sum((p[i]-cost[i])*exp(valuation[i]-p
[i])/normp for i = 1:n))

@NLconstraint(m,1+sum(exp(valuation[i]-p[i]) for i = 1:n)
==normp)

print(m)

solve(m)

m1= Model(solver=IpoptSolver())

@variable(m1,x[0:n]>=0)

@NLobjective(m1,Max,(1-x[0])*log(x[0])+sum((valuation[i]-log
(x[i])-cost[i])*x[i] for i = 1:n))

@constraint(m1,sum(x)==1)

print(m1)

solve(m1)

getvalue(x)

for i = 1:n
   println(valuation[i]-log(getvalue(x[i]))+log(getvalue(x
[0])))
end

getobjectivevalue(m1)

optprice = [46.476695956178155
47.476695898156
45.4766959736541
44.476695919675976
47.47669597286191
43.47669591320686
46.47669594742878
```

```
47.47662854942838
46.47647227448389
41.476604968547825
36.47644877605803
44.47642281269446
42.47644877605803
46.47647227448389
45.47669262288309
42.47665077531242
44.47668589262447
47.47669044434047
42.47668685263434
45.476694745722604]

Iter = 100000
randomprofit = zeros(Iter)
srand(2017)
for k = 1:Iter
    p = baseprice + 5*(2*rand(n,1)-1)
    randomprofit[k] = dot((p-cost),(exp.(valuation-p)/(1+sum
(exp.(valuation-p)))))
end

maximum(randomprofit)
```

The Analytics Edge

There are several attributes of revenue management systems that make them widely used:

1) Finds ways to increase revenue without necessarily changing products & companies
2) Product prices are more aligned to what customers are willing to pay.
3) Computational power has made it possible to solve complex optimization models & reoptimize based on realized demand & implement strategies in real time.

Capstone allocation

Tool: Optimization (integer)

The Analytics Edge

The launch of a capstone program in the university needs students to be allocated. Some of the main challenges are:

1) **Fairness**: What does a fair allocation of projects to students mean? This is often subjective & a challenge is to identify this

2) **Efficiency**: It is important that the capstone allocation is efficient in taking account of heterogeneous student preferences

3) **Flexible**: The allocation should be flexible to account for various constraints from stakeholders - students, faculty, capstone committee

4) **Multidisciplinarity**: The projects need to have a mix of students from disciplines.

```
using JuMP, Gurobi, DataFrames

Rank = readtable("Rank.csv",header=false)

size(Rank)

Pillar = readtable("Pillar.csv",header=false)

size(Pillar)

LowerBound = readtable("LowerBound.csv",header=false)

size(LowerBound)

UpperBound = readtable("UpperBound.csv",header=false)

size(UpperBound)

Students = 1:170

Projects = 1:61

PillarId = 1:4

m = Model(solver=GurobiSolver())

@variable(m, x[Students,Projects], Bin)

@variable(m, y[Projects], Bin)

@constraint(m, allocatestudent[i = Students], sum(x[i,j] for
j = Projects)==1)

@constraint(m, allocateifprojectoffered[i = Students, j =
Projects], x[i,j] <= y[j])

@constraint(m, lower[j = Projects, k = PillarId], sum(x[i,j]
for i = Students if Pillar[i,k]==1) >= LowerBound[k,j]*y[j])

@constraint(m, upper[j=Projects,k=PillarId],sum(x[i,j]*Pillar
[i,k] for i = Students) <= UpperBound[k,j]*y[j])

@constraint(m, allocateonlyifranked
[i=Students,j=Projects;Rank[i,j]==0],x[i,j]==0)

@constraint(m, lowerprojectsize[j=Projects],sum(x[i,j] for
i=Students)>=5*y[j])

@constraint(m, upperprojectsize[j=Projects],sum(x[i,j] for
i=Students)<=8*y[j])

@objective(m, Max, sum(Rank[i,j]*x[i,j] for i = Students, j =
Projects))

@objective(m, Max, sum(Rank[i,j]*x[i,j] for i = Students, j =
Projects))
```

```
solve(m)

getobjectivevalue(m)

getvalue(y)

getvalue(x)

getvalue(x[1,1])

for i = Students
    for j = Projects
        if(getvalue(x[i,j])==1)
            println("Student ", i, " Project ", j)
        end
    end
end

FinalRankedProject=zeros(10)
for i = Students
    for j = Projects
        if(getvalue(x[i,j])==1)
            FinalRankedProject[Rank[i,j]] =
FinalRankedProject[Rank[i,j]]+1
        end
    end
end

FinalRankedProject

@constraint(m,sum(Rank[i,j]*x[i,j] for i = Students, j =
Projects) == getobjectivevalue(m))

@objective(m, Min, sum(x[i,j] for i = Students, j = Projects
if Rank[i,j] == 1))
solve(m)

for t = 1:9
@objective(m, Min, sum(x[i,j] for i = Students, j = Projects
if Rank[i,j] == t))
solve(m)
@constraint(m, sum(x[i,j] for i = Students, j = Projects if
Rank[i,j] == t) == getobjectivevalue(m))
end

FinalRankedProject=zeros(10)
for i = Students
    for j = Projects
        if(getvalue(x[i,j])==1)
            FinalRankedProject[Rank[i,j]] =
FinalRankedProject[Rank[i,j]]+1
        end
    end
end

FinalRankedProject
```