Summer Internship Report

April 21, 2020 – July 17, 2020

University of Concordia, Montreal

John Molson School of business

Topic: Retail Demand Management: Forecasting, Assortment Planning and Pricing

Vishanth Hari Raj B
Production and Industrial Engineering
National Institute of Technology, Tiruchirapalli

ABSTRACT:

In the first part of the research project, the focus is on the retailer's problem of optimizing the selected assortment of the products and customizing the assortment of a store to maximize revenues or profits. We develop heuristic algorithms for assortment optimization in reference to various algorithms like Greedy algorithm, Hill climbing algorithm etc. and demonstrate it uses in practical applications. We model a retailer's assortment planning problem under a ranking-based choice model of consumer preferences. Under this consumer choice model each customer belongs to a type, where a type is a ranking of the potential products by the order of preference, and the customer purchases his highest ranked product (if any) offered in the assortment. In the model we consider products with different price/cost parameters, we assume that the retailer incurs a fixed carrying cost per product offered, a substitution penalty for each customer who does not purchase his first choice and a lost sale penalty cost for each customer who leaves the store empty-handed. In the absence of any restrictions on the consumer types, searching for the optimal assortment using enumeration or integer programming is not practically feasible. We developed an optimal assortment has structure like greedy-type heuristics to find the optimal assortment and maximize the profit.

CONTENTS

Introduction:

In the research Internship, the main objective developed a stylized choice model in which consumers preferences are defined based on their ranking of products that could potentially be included in the assortment. We study a retailer's product selection process under this ranking-based consumer choice Model in the presence of fixed, substitution, and lost sale penalty costs, investigate the existence of a simple structure for the optimal assortment, examine the performance of greedy-type heuristics, and propose an effective algorithm to solve the problem optimally. Under the Ranking-based consumer choice model enumerating all possible assortments or using integer programming are practically infeasible methods when the number of potential products to choose from is large. Because the problem is NP-hard, an optimal assortment cannot generally be obtained by ranking products in terms of popularity or profitability. We develop a new algorithm which always obtains an optimal assortment. This algorithm rests on an iterative procedure which at each step includes/exclude products that improve/worsen the profitability of every assortment.

In the first part of the research project, the focus is on the retailer's problem of optimizing the selected assortment of the products and customizing the assortment of a store to maximize revenues or profits. We develop heuristic algorithms for assortment optimization in reference to various algorithms like Greedy algorithm, Hill climbing algorithm etc. and demonstrate it uses in practical applications. We model a retailer's assortment planning problem under a ranking-based choice model of consumer preferences. Under this consumer choice model each customer belongs to a type, where a type is a ranking of the potential products by the order of preference, and the customer purchases his highest ranked product (if any) offered in the assortment. In the model we consider products with different price/cost parameters, we assume that the retailer incurs a fixed carrying cost per product offered, a substitution penalty for each customer who does not purchase his first choice and a lost sale penalty cost for each customer who leaves the store empty-handed. In the absence of any restrictions on the consumer types, searching for the optimal assortment using enumeration or integer programming is not practically feasible. We developed an optimal assortment has structure like greedy-type heuristics to find the optimal assortment and maximize the profit.

Literature Review:

Retail demand management: forecasting, assortment planning and pricing:

Referred various Research Paper on the topic of Retail Demand management- Assortment planning and pricing

Consumer Choice Modelling:

Consumer choice models constitute the fundamental platform for assortment planning, and may be classified as
(1) utility-based models
(2) exogenous demand models.

Utility Based Models:

Utility based choice models assume that every customer associates a utility U_i with each product $i \in N$. In addition, there is a no-purchase option denoted $i = 0$, with associated utility U_0. When offered an assortment S, every customer chooses the option giving him the highest utility in $S \in \{0\}$. The market share for each SKU $i \in S$ can then be evaluated once we know the distribution of utilities across the consumer population.

The Multinomial Logit (MNL) model is the most extensively studied utility-based model in the marketing and economics literature. The MNL model assumes that the utilities U_i can be decomposed into a deterministic component U_i that represents the average utility derived by the population of customers, and a random component X_i that represents idiosyncrasies across customers. The X_i are assumed to be identical and independent Gumbel random variables with mean zero and scale parameter m.

The other commonly used utility-based model is the Locational Choice model to study pricing and store location decisions of competing firms. Lancaster (1966, 1975) extended this work to a locational model of consumer product choice, where every SKU $i \in N$ is represented as a bundles of attribute levels $z_i = (z_{1i}, z_{2i}, ..., z_{Ti}) \in R_T$

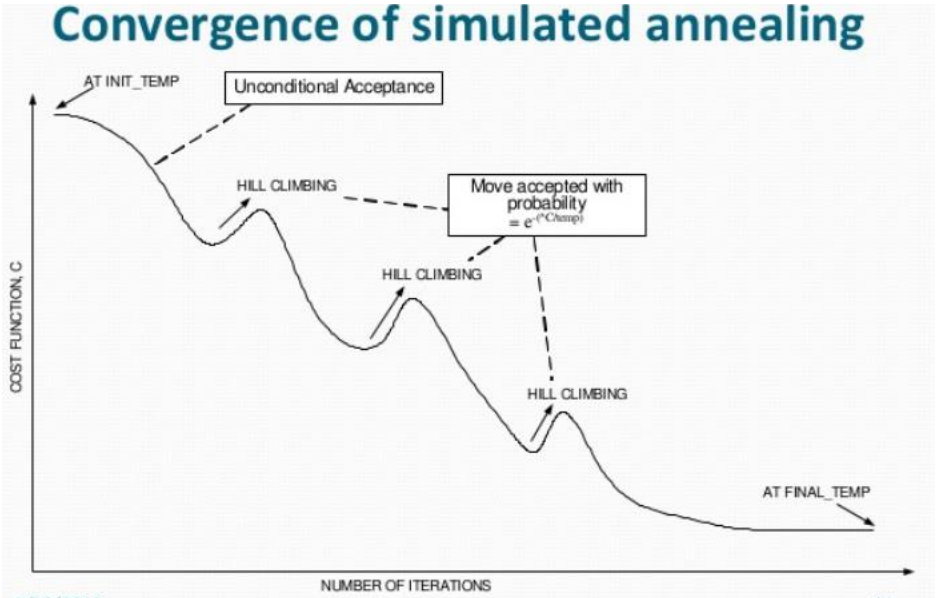
In the exogenous demand model:

Every consumer is assumed to have a favourite product *i*, and f_i is the share of consumers whose favourite product is *i*. A consumer whose favourite product is *i* buys it if $i \in S$; if $i \in S$ they substitute to SKU $j \in S$ with probability a_i. The exogenous demand model has more degrees of freedom than the MNL and LC models and can accommodate extremely flexible substitution structures.

Hill Climbing Algorithm/ Simulated Annealing Algorithm and Tabu Search Algorithm:

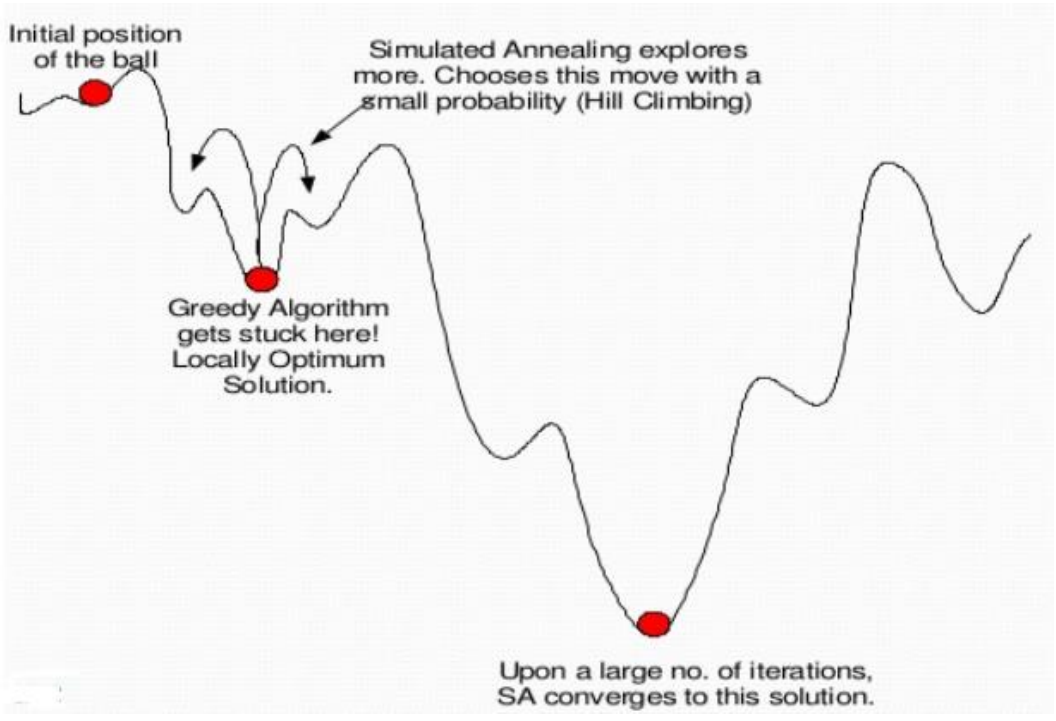
Simulated annealing maintains a current assignment of values to variables. At each step, it picks a variable at random, then picks a value at random. If assigning that value to the variable is an improvement or does not increase the number of conflicts, the algorithm accepts the assignment and there is a new current assignment. Otherwise, it accepts the assignment with some probability, depending on the temperature and how much worse it is than the current assignment. If the change is not accepted, the current assignment is unchanged.
To control how many worsening steps are accepted, there is a positive real-valued temperature T. Suppose A is the current assignment of a value to each variable. Suppose that h(A) is the evaluation of assignment A to be minimized. For solving constraints, h is typically the number of conflicts. Simulated annealing selects a neighbour at random, which gives a new assignment A'. If $h(A') \leq h(A)$, it accepts the assignment and A' becomes the new assignment. Otherwise, the assignment is only accepted randomly with probability

$e^{(h(A)-h(A'))/T}$.



Thus, if h(A') is close to h(A), the assignment is more likely to be accepted. If the temperature is high, the exponent will be close to zero, and so the probability will be close to 1. As the temperature approaches zero, the exponent approaches -∞, and the probability approaches zero. With number of iterations increasing the more efficient solution can be found but this might lead to high computing time due to high number of iterations. It becomes necessary to optimise between computing time (number of iteration) and in reaching optimal solution.

On easy comparison of greedy algorithm and simulated Annealing Algorithm is depict in the below figure. Greedy algorithm often stuck in global maxima/minima which restricts the algorithm in finding the optimal solution. With the large number of iterations, the optimal solution can be obtained but the solution to Assortment problem should be simple and less complex, should take less computing time. So, in order to overcome the problem a hybrid combination of Greedy and Simulated Annealing Algorithm is created and testing its effectiveness in find the solution.



In nonlinear problems, It becomes necessary to create a heuristics Algorithm for finding the best solution rather than wasting the time in finding optimal solution. A wide range of review on heuristics Algorithm has been done which includes Hill climbing Algorithm, Tabu search which aims to find the best solution in Non-Linear Problem.

Problem Formulation/ Statement:

The problem statement is to develop a Greedy-based heuristics algorithm to find the optimum Assortment S* with maximizing the profit of a Store with compared less time complexity (less computing time). Profit specified equation should be considered the Lost sales Penalties and substitution parameters and penalty cost in-order to estimate closets practical profits in the store.

Development of Model:

The Model is based on how the consumers make choices. Consider a product category consisting of n potential products, indexed *1 to n*, where $N \epsilon \{1,\ldots..,n\}$ and let 0 denote the no-purchase option. In order to define the purchase behaviour of consumers. Let K be the maximum number of Products in an assortment in a store. Let L be the Maximum no of different Assortment.

Notation Description

N	**Index set of all possible Products a retailer could carry in this category**
K	**Maximum number of Product quantities SKUs per assortment**
L	**Maximum number of different assortments**
P_i^k	**Proportion of customer having i^{th} product has kth choice**
Π_i	**Profit of the i^{th} product**
f(.)	**Substitution Penalty Function**
p_i	**lost sale penalty cost of i^{th} item**
X_i	**Stock-out product i^{th} product.**
S*	**Optimum Assortment (maximum profit)**
S	**Generic assortment type.**

Let j denotes the profit margin on a product j. Retailers generally incur a fixed cost, denoted by the parameter K, of carrying a product in their assortment. In addition, we assume that the retailer bears a substitution penalty cost for every customer who purchases a product that is not his first choice, and that the penalty is higher when he purchases a lower ranked product. More specifically, if a customer buys his k^{th} favourite product, the penalty is f(k) where f is an non decreasing function, e.g., $f(k) = b(k-1)$ where $b < 0$.We do not make any assumption about $f(k)$, so it is possible that $j \epsilon f(k) < 0$ for some j; $k \epsilon f_1,\ldots f_N$, meaning that the retailer may lose money when a customer buys product j as a k^{th} choice. The value of substitution penalty value is subtracted from the customer demand proportion.

Substitution penalty function:

f(k) = b*(k -1) ***where b = 0.2 (constant value) k is choice no.***

The lost sale penalty is defined as parameter which denotes the loss of sale value due to customer doesn't find the preferred product as the result , that store lost some amount of profit because of that type of assortment S and also proportional of customer visit that store and end up in not buy anything because of various reason. We defined lost penalty function in which the overall lost sale penalty function is a quadratic polynomial function depend on stock-out products, which directly indicates lost product sales, as customer didn't buy anything as their preferred wasn't available in the store. Also let p_i denote a lost sale penalty cost which measures the disutility experienced by a customer who leaves the store empty-handed. Virtual penalty is defined as another parameter to includes the effects of sale which has been affected due to loss of customer's dissatisfaction. It indicates that profit value reduction which had happen to present customer' dissatisfaction which might lead to overall popularity of the store or which might lead to negatives view due to which the some proportion of sale might have lost which lead to reduction profit because of that particular assortment S.

Lost sale penalty plus Virtual penalty function:

$$F_i(x) = p_i * x + (p_i * x^2)/2$$

p_i – Lost sale penalty value of i^{th} product

X_i – Stock out value of i^{th} product

K_0 is fixed cost which has been add to every assortment type, which include miscellaneous charges. The profit of the assortment is calculated using the given following formulae. Which directly multiples with the profit of the product with person's first choice of interest and with addition to profit of the product is multiplied with person's second choice if the quantities of his first choice is out of the stock, with subtraction of respective substitution penalty function (which is depended on customer choice) and similar the process is repeated up to k^{th} choice of the proportion of the customer, And the common fixed cost is subtracted from the total profit π of the Assortment S. Later, from the total profit of that assortment, Lost penalty value (which is directly calculated from the quadratic polynomial equation that is dependent on Stock-Out value $X_i \epsilon N$ and it is related) is subtracted which be the total estimated profit of that assortment.

Total Profit formula:

$$\Pi_S^* = \sum_{i=0}^{n-1} \sum_{k=1}^{n} P_i^k(s)^* [\pi_i - f(k)] - F_i$$

Overall Profit of an Assortment S is calculated using the above formula. The derivation of formula is obtained by simple mathematical calculation which has been explained in following steps:

Step 1: The multiplication of $P_i^k(s)$ square matrix (as *i=k* where k is the choice of customer ***k= 1,2….,n*** similarly ***i= 1,2,…,n)*** with the profit of the product π_i .

Step 2: Calculation of substitution penalty which is obtained by multiplication of $P_i^k(s)$ square matrix with ***f(k)*** and the result value of the product.

Step 3: Calculation of Virtual penalty and lost sale penalty of each product and stored in array which can be calculated by the given formula $F_i(x) = p_i * x + (p_i * x^2)/2$

Step 4: Finally, the substitution penalty and virtual penalty of the product is subtracted from initial value obtained from *Step 1*.

Step 5: Repeat the above process from *step 1- step 5* for all the products ***i=1,2,…,n*** in the Assortment S.

Step 6: Adding up the overall values of each product profit in the *Assortment* ***S*** gives up the overall profit of that Assortment ***S***.

The overall simplified mathematical expression for the calculation of optimal profit in Assortment S in the model is given below.

$$\Pi(S^*) = \max_{S \subseteq \mathcal{N}} \Pi(S).$$

Heuristics Algorithm for Optimal Assortments:

If there were no substitution, then the assortment problem could be optimally solved by a greedy algorithm that chose products in decreasing order of their profit contribution. But substitution makes the objective function nonlinear because overall profit of the assortment depends in part on its substitution penalty and Lost sales penalty function, which depends on which Stock-out value of the products are in the assortment. We developed algorithm which can be classified into two sub algorithms in order to estimate the best valued profit in given time and less complexity.

First Part of algorithm:

It involves the calculation of maximum profit contributed by each product type with the subtraction of each product substitutional penalty value and lost sale penalty / virtual penalty due to various in consumer's choices, customer going in empty handed and the loss of customer respectively. With these each product profit weights and quantity of the products that are available in the inventory/Store, the most profitable assortment is created in the manner of Greedy based-approach which sorts the quantities of products in the assortment purely based on decreasing order of the Profits of the products, i.e The highest profit products comes at first preference, second high profit product is preferred next and it goes on till the it satisfies the total no of products quantities that can be kept in the assortment N .

Algorithm:

```
Initialize i=0;
While (i<n)                n is the total number of product items
  {
    used[i] = 0;       (initially assigning value to used[] array)
  }
  curw = w;
  //loop until knapsack is full
  while (curw >= 0)
  {
    maxi = -1;
    //Create a loop to find max profit object.
    for (i = 0; i < n; i++)
    {
      if ((used[i] == 0) && ((maxi == -1) || (((float) array[1][i]
          / (float) array[0][i]) > ((float) array[1][maxi]
```

```
            / (float) array[0][maxi])))
        {
           maxi = i;
        }
      }
      used[maxi] = 1;
      //decrease current wight
      curw -= array[0][maxi];
      //increase total profit
      totalprofit += array[1][maxi];
      if (curw >= 0)
      {
        print "\nAdded item " << maxi + 1 << " quantity "
            << array[0][maxi] << " revenue: " << array[1][maxi]
            << " completely in the Store, Space left: " << curw;
      k= maxi+1;
       newarray[0][k-1]= array[0][maxi]; //quantity of new arranged value
       newarray[1][k-1]= array[1][maxi];  // revenue new type assortment

      }
      else
      {
        print "\nAdded item " << maxi + 1 << " Quantity: "
            << (array[0][maxi] + curw) << " Revenue: "
            << (array[1][maxi] / array[0][maxi]) * (array[0][maxi]
                + curw) << " partially in the store, item Space left: 0"
            << " Item added is: " << curw + array[0][maxi];
        totalprofit -= array[1][maxi];
        totalprofit += ((array[1][maxi] / array[0][maxi]) * (array[0][maxi]
            + curw));
        k= maxi+1;
        newarray[0][k-1]=(array[0][maxi] + curw); //quantity of new arranged value
        newarray[1][k-1]=  (array[1][maxi] / array[0][maxi]) * (array[0][maxi]+curw);// revenue new type
assortment
      }
   }
   //print total worth of objects filled in knapsack
   print "\n Store filled with items worth: " << totalprofit;
```

Second -Part of the Algorithm:

Second- Part of the algorithm is derived from the basis of hill climbing algorithm. Hill climbing algorithm aims to find the highest profit value with the given initial value point in non-linear problems by estimating the next point and avoid the drench created by local minima or local maxima. Same concept has been inherited in this part of the algorithm considering the Assortment S which has been obtained from the first part greedy based approach as the starting point in non-linear function and estimating the most profit assortment type from that which reduces the time complexity by not analysing the all the point (type of assortment combination)

The second part of algorithm involves, checking the current assortment (which as assumed to starting point of the assortment) is optimal or not by decreasing the quantities of product which has high lost sale/virtual penalty value and also by increasing the same amount of quantities to the product which has lower penalty value due the various in stock-out value which has direct dependence on quantities of that product in the assortment. During the iteration of above process, the closely optimum assortment can be identified, which has maximum profit from the rest type of the assortment. Iteration Limit can be fixed depending upon the problem in order allow less iteration and checking of the unnecessary assortment type. As sometimes the profit might decrease, increase and then decrease and increase because of local maxima, it might stick into the local maxima which further reduces efficiency of the algorithm. In-order to overcome the problem particular optimum number of iteration is provide so that it will be enough to not to get drenched by local maxima and also doesn't increase the time complexity and computing of algorithm by reduce the search of unnecessary point in non-linear assortment profit function.

Algorithm:

```
do
{
float penfuncd[1000]; //initilzing the dumpy penalty fun to sort out the orginal list of penalty func values
int minipen=penfunc[0]; //finidng the minimum penalty

int maxstock=0;//determines the item which has maximum stock-out (finding the minimum penalty)
int a=0,s=0;
for(int i=0;i<b;i++)  //assigning value to dummy penfunc
{
```

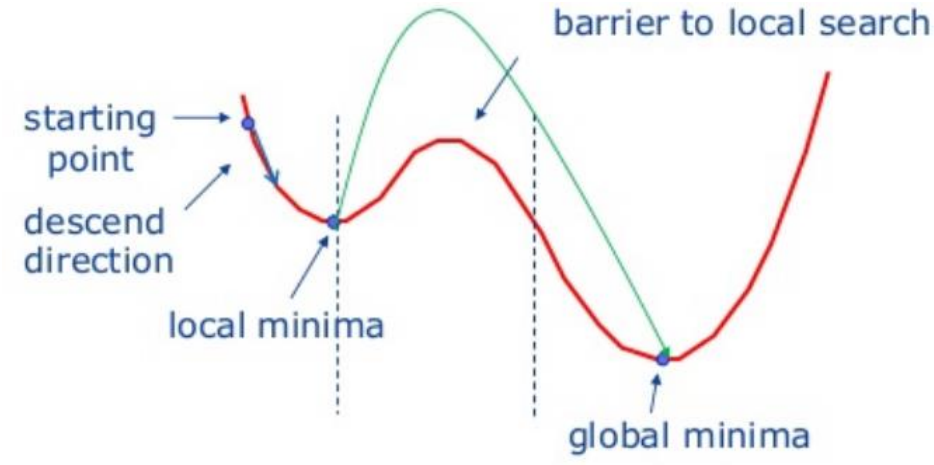
```
        penfuncd[i]=penfunc[i];
        cout<<"\n Penalty value "<<penfunc[i];
        cout<<"\n revenue value "<<revenue[i];
        cout<<"\n quantity in assortment and their revenue
"<<newarray[0][i]<<"\t"<<newarray[1][i];
}
sort(penfuncd,penfuncd+b); //sorting in order to find the highest and lower penalty fun
value and also it's pos
for(int i=0;i<b;i++)
{
        cout<<"\norder penalty"<<penfuncd[i];
}
for(int i=0;i<b;i++)
{
        if(penfuncd[0]==penfunc[i])
        {
                a=i;

        }
}
cout<<"\n a value"<<a;
if(newarray[0][a]>0) //checking whether the quantity of the corresponding item in
assortment is greater than zero to avoid negative values
{
newarray[0][a]=newarray[0][a]-1;
//cout<<"\nif this loop exec";
}
else    //finding the second minimum penalty in which quantity in assortment won't be 0
alternative way of finding second biggest penalty value
{
//cout<<"\n else loop exec";
 for(int i=0;i<b;i++)
{
        if(penfuncd[1]==penfunc[i])
        {
                a=i;
        }
}
   newarray[0][a]=newarray[0][a]-1;  }
```

```
//ends the part of program which finds the second lowest penalty


P[a]=P[a]+1;    //stock-out is increases as quantity is assortment is decreased by one
penfunc[a]=(P[a]*penality[a])+((pow(P[a],1.5)*penality[a])/2);  //revised penalty function
values

revenue[a]= A[a]*brand[a].j0-penfunc[a];//revising the revenue valuess

for(int i=0;i<b;i++)   //find the maximum penalty function value (maxstock implies max
penalty)
{
        if (penfunc[i]==penfuncd[b-1])
        {
    s=i;
                }
}
P[s]=P[s]-1;
newarray[0][s]=newarray[0][s]+1;
penfunc[s]=(P[s]*penality[s])+((pow(P[s],1.5)*penality[s])/2);
revenue[s]= A[s]*brand[s].j0-penfunc[s];

revisedprofit = revisedprofit+((revenue[s]/brand[s].j0))-((revenue[a]/brand[a].j0));
cout<<"\n LOOP REVISED PROFIT"<<revisedprofit;
for(int i=0;i<b;i++)
{
cout<<"\n Item quantity for each product for "<<i+1<<"\t is \t"<<newarray[0][i]<<"\n profit
made by the product ";
cout<<"\tEnter the decide=1 to continue or 0 to exit\t";
cin>>decide;
}while(decide==1);
```

The second part of the algorithm iterates each time to verify, whether the profit can be maximized by changing the assortment type. The number iteration can be fixed by the person depending upon one's interest on balancing computing time with difference between predicted profit and optimal profit. As of this project Iteration is fixed as 30 iteration for study purpose of this algorithm on effective it has predicted the overall estimated profit.

Analysis of the Algorithm:

Algorithm mainly of a testing purpose to find how close the estimated profit with the optimal profit. Majorly virtual penalty function power is fixed and predicted in trial and error basis on how much effect it has on the total profit value. Several Existing algorithms generally finds the value of local minima thus by restricting to advance to find the global maximum, optimal value as given in the below figure. The algorithm advance the next step using the analysis of each product which has highest and lowest overall virtual penalty, which is function of stock-out products, Algorithm thereby proceed forward by reducing stock-out product of highest by replacing that quantity with the product having lowest virtual penalty value. In this way the algorithm procced to find the next assortment type in each iteration of the program.



Certain assumption has been done in the algorithm; function of virtual penalty is quadratic function. On trial and error basis the power of the function is finalised, it can also be varied depending on the actual practical purpose. Various virtual penalty had been discussed during the sessions, the graph nature of all function has been reviewed and it mainly aligns and resembles with polynomial. As virtual penalty must be increasing when the stock-out of the product increasing, it leads to dissatisfaction of customer which might lead to negative reviews and less popularity, on identifying the dissatisfaction of customer and it negative impact on the popularity which will increase exponential power which might have high effect if it reaches to certain threshold limit. In these idea of analysis, virtual penalty function is assumed to quadratic equation.

Sample Problem 1:

The given problem with variables values is given below in the table.

	Quantity of product	Profit of the product per q	Porportions of custome	Porportions of custon	Porportions of cust	Lost Sale Value	stock out value (initial)	
Product 1	4	200	0.5	0.3	0.1	10	6	
Product 2	5	120	0.7	0.4	0.25	8	8	
Product 3	6	100	0.8	0.5	0.1	5	5	

Now, Algorithm aims to achieve an Assortment where the overall profit. As it is heuristics algorithm which is bounded by greedy type algorithm and a nonlinear programming estimates the profit which nearness of optimal profit. We will analysis it results and computing capacity with each iteration to decide the near optimal profit. It also identifies the assortment which also given highest profit.

Output:

```
Added item 1 quantity 4 revenue: 584.015 completely in the Store, Space left: 6
Added item 3 quantity 6 revenue: 781.799 completely in the Store, Space left: 0
Added item 2 Quantity: 0 Revenue: 0 partially in the store, item Space left: 0 Item added is: 0
 Store filled with items worth: 1365.81
 Item    1 Its quantity and profit share in assortment  4
 Item    2 Its quantity and profit share in assortment  0
 Item    3 Its quantity and profit share in assortment  6
 Penalty value 133.485
 revenue value 584.015
 quantity in assortment and their revenue 4     584.015
 Penalty value 154.51
 revenue value 649.865
 quantity in assortment and their revenue 0     0
 Penalty value 52.9509
 revenue value 781.799
 quantity in assortment and their revenue 6     781.799
order penalty52.9509
order penalty133.485
order penalty154.51
 a value2
 LOOP REVISED PROFIT1372.67
 Item quantity for each product for 1    is     4
 profit made by the product
 Item quantity for each product for 2    is     1
 profit made by the product
 Item quantity for each product for 3    is     5
 profit made by the product     Enter the decide=1 to continue or 0 to exit     1

 Penalty value 133.485
 revenue value 584.015
 quantity in assortment and their revenue 4     584.015
 Penalty value 130.081
 revenue value 674.294
 quantity in assortment and their revenue 1     0
 Penalty value 66.7423
 revenue value 768.008
 quantity in assortment and their revenue 5     781.799
order penalty66.7423
order penalty130.081
order penalty133.485
 a value2
 LOOP REVISED PROFIT1400
 Item quantity for each product for 1    is     5
 profit made by the product
 Item quantity for each product for 2    is     1
 profit made by the product
 Item quantity for each product for 3    is     4
 profit made by the product     Enter the decide=1 to continue or 0 to exit     1
```

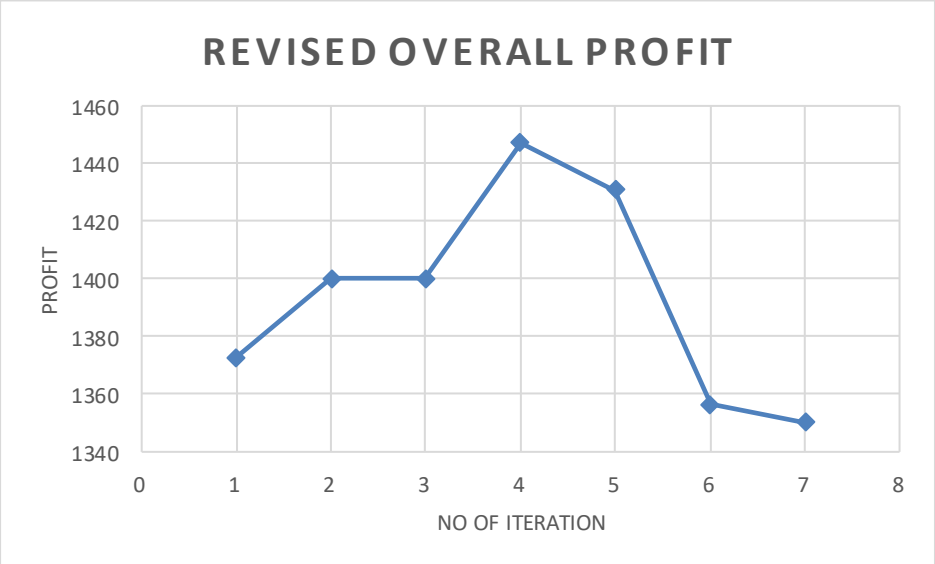
Output of the algorithm has been shown in the above figures. From the each output, penalty values are modified for each iteration, checking its effects on overall profit, one of two major objectives is to reduce overall virtual penalty value by changing the stock-out value in the assortment and another one is maximize the overall profit.

Revised Overall Profit are given below for each iteration. The highlighted part shows the maximized profit among them. We also investigated the graph for easy understanding and nature on how the overall profit is changed according to each iteration (a possible assortment combination from the total number of possible combination).

Table 1:

Number of Iterations	Revised Overall Profit
0	1365.81
1	1372.67
2	1400
3	1400
4	1447.11
5	1430.62
6	1356.2
7	1350.12

Graph 1:



From the graph, we can understand the overall profit at each instance of the iteration, which helps to find the optimal Assortment and profit, thus by not being deadlocked in local maximum.

Problem 2:

	Quantity	Profit c	Porportions of custom	Porportior	Porportior	Lost Sale Valu	stock out value (initial
Product 1	12	200	0.5	0.3	0.1	13	13
Product 2	20	120	0.7	0.4	0.25	14	14
Product 3	18	100	0.8	0.5	0.1	18	16

Output:

Similarly, to first sample problem, the value and certain modification is done to check its versatility of the algorithm and its effectiveness in finding optimal solution.

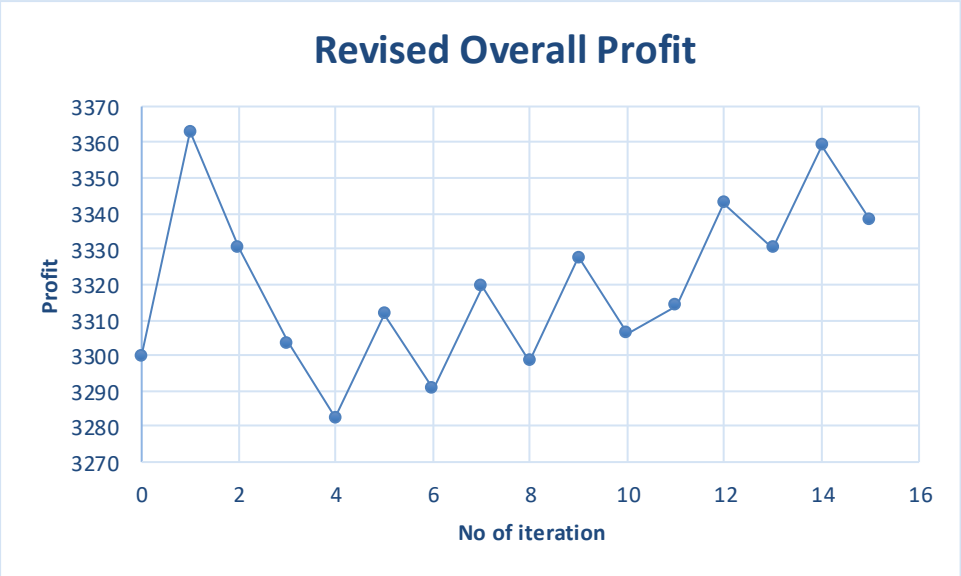
```
dded item 1 quantity 12 revenue: 1678.83 completely in the Store, Space left: 13
dded item 2 Quantity: 13 Revenue: 1725.63 partially in the store, item Space left: 0 Item added is: 13
Store filled with items worth: 3404.46
Item    1 Its quantity and profit share in assortment  12
Item    2 Its quantity and profit share in assortment  13
Item    3 Its quantity and profit share in assortment  0
Penalty value 473.669
revenue value 1678.83
quantity in assortment and their revenue 12    1678.83
Penalty value 562.682
revenue value 2654.82
quantity in assortment and their revenue 13    1725.63
Penalty value 864
revenue value 1640.25
quantity in assortment and their revenue 0     0
rder penalty473.669
rder penalty562.682
rder penalty864
a value0
LOOP REVISED PROFIT3363.71
Item quantity for each product for 1    is     11
profit made by the product
Item quantity for each product for 2    is     13
profit made by the product
Item quantity for each product for 3    is     1
profit made by the product     Enter the decide=1 to continue or 0 to exit     1

Penalty value 522.491
revenue value 1630.01
quantity in assortment and their revenue 11    1678.83
Penalty value 562.682
revenue value 2654.82
quantity in assortment and their revenue 13    1725.63
Penalty value 792.853
revenue value 1711.4
quantity in assortment and their revenue 1     0
rder penalty522.491
rder penalty562.682
rder penalty792.853
a value0
LOOP REVISED PROFIT3330.98
```

The Above figures shows upto only first two iteration. Below tables denote the overall profit at the end fo each iteration.

Table 2:

Number of Iterations	Revised Overall Profit
0	3300
1	3363
2	3330.14
3	3303.63
4	3282.56
5	3311.56
6	3290.53
7	3319.5
8	3298.46
9	3327.43
10	3306.39
11	3314
12	3343
13	3330.19
14	3359.16
15	3338.13

Graph 2:



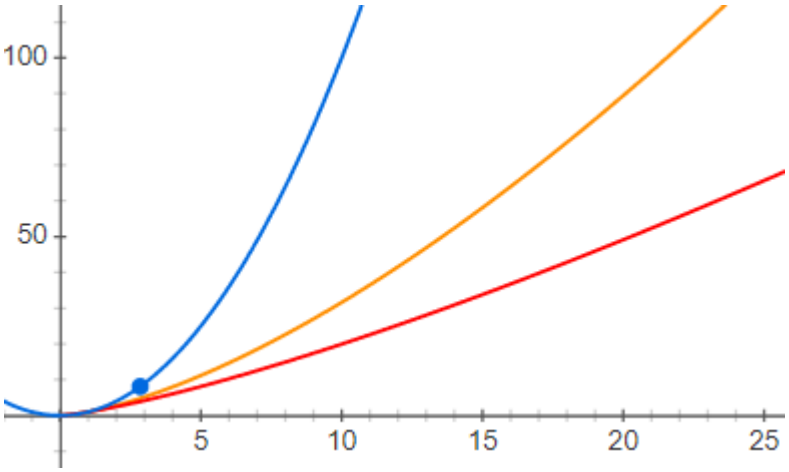
From the graph, we can understand the overall profit at each instance of the iteration, which helps to find the near optimal Assortment. With the number of iterations increases, we can able to confirm how accurate it can give best solution near to optimal solution. Starting point of Iteration (starting point of Assortment S) is found out using Greedy Algorithm and it is improvised using the concept of Hill climbing Algorithm.

Robustness study of the Model:

In all cases, the methodology of algorithm is as follows. We first calculate the assortment Sr which is optimal for a given set of parameter values called the base set (the optimal solution is obtained using the Greedy Algorithm). The base set represents the values used by the firm, which may not be the correct ones. Then we vary some of the parameters (one at a time) around the base set. These numbers represent the `correct' parameter values. Using these numbers, we compute the optimal profit and the profit of assortment Sr under the correct parameters. Then we calculate the percentage optimality gap of assortment Sr, which measures the percentage loss in profit from not using the correct values of the parameters.

Analysis of power of virtual penalty function.

$F_1(x) = x^2$
$F_2(x) = x^{1.5}$
$F_3(x) = x^{1.3}$



Among the function analysed, it depends on the variation of other aesthetics and non-materialistic like existing popularity of store, type of people lives in the location and degree of localisation. Similar analysis is done for substitution penalty function which proves to be effective function and extensively proposed by other researchers in different journals and research paper.

Conclusion:

In this dissertation, we have mainly focused on three issues affecting retail demand management. First, we addressed the issue of demand estimation, assortment optimization and assortment localization in product retailing. Second, the investigated is done on the sensitivity of the optimal assortment and expected profits on the key assumptions made about the choice model and substitution under stock-outs.

In this project, we formulated a process for finding optimal assortments, comprised of a demand model, estimation approach and heuristics for choosing assortments. The process is applied to sample data and showed that the approach produces accurate forecasts for the given problem. The analysis of algorithm is done, on comparing it variation to different problem model and its efficiency and effectiveness in finding the best solution. The investigation on impact of parameters and commonly made assumption has been done.

In theory the Developed heuristics Algorithm has the same complexity as the enumeration method but in practice it is much faster than other Nonlinear algorithms, for a problem with 20-30 products. Based on an extensive numerical analysis developed Algorithm provide best optimal solution for problems with up to 25 to 30 products.

Some limiting assumption is the that of stock-out based substitution which obviates the need for considering stock levels of the product. Under stock-out based substitution, the demand for a product not only depends on the initial assortment, but also on the inventory level of the other products.

Bibliography

- Retail Demand Management: Forecasting, Assortment Planning and Pricing: Ramnath Vaidyanathan University of Pennsylvania

- Besbes, O., D. Saure. 2010. Product assortment and price competition under multinomial logit demand.

- An Inventory-Theoretic Approach to Product Assortment and Shelf-Space Allocation TIMOTHY L. URBAN, University of Tulsa

- Forklift Routing Optimization in a Warehouse using a Clustering-based Approach Rahman, Saif Muhammad Musfir (2019) Forklift Routing Optimization in a Warehouse using a Clustering-based Approach.

- Cachon, G., C. Terwiesch. 2008. Matching Supply with Demand: An Introduction to Operations Management. McGraw-Hill, Boston

- Cooperation between two suppliers and a common retailer salimi, saba (2015)

- https://www.cs.ubc.ca/~poole/aibook/html/ArtInt_89.html - Simulated Annealing Research works

- https://github.com/ GitHub Repository (For reference of various code and greedy algorithms)