Designing an Expert System for Online Shopping Cart Management

Deepshikha Bhargava[1], Pratikshya Mishra[2], Anjali Mishra[3]

[1]Professor, University of Petroleum and Energy Studies, Dehradun, India
[2,3]M.Tech CSE, University of Petroleum and Energy Studies, Dehradun, India
[1]deepshikhabhargava@gmail.com, [2]pratikshya.mishra72@gmail.com, [3]anjaliannu100@gmail.com

Abstract: In this digital era consumers prefer online shopping to save time. But in their busy schedules it can be time-consuming to choose items as per their requirements. There is a need to have a smart shopping cart that gets filled with products as per the customer's criteria in lesser time. To adhere this issue, this paper proposed an expert system where the shopping cart management is considered as a Knapsack Problem. The Shopping Cart Management is considered as a 0/1 Knapsack Problem, where the problem is to prepare an optimized shopping cart as per the customer requirement bounded with the budget. This problem is solved through the procedure based Expert System which uses the Dynamic Programming approach.

Keywords: dynamic programming, 0/1 Knapsack Problem, e-commerce website, expert systems

I. INTRODUCTION

A. The 0/1 Knapsack Problem

The knapsack problem is one of the optimization problems. The objective is to maximize the number of elements that can be filled in one bag of a certain weight such that it results in maximum value. From among a group of items, each having a cost and a value associated with it, we estimate which all items to be included in a collection so that the total value is maximum and the total cost is less than the given cost.[1][10]

The decisive question in the knapsack problem is whether a maximum value of at least V can be achieved while the cost is less than or equal to C.

Fig. 1. Knapsack Problem

In the knapsack problem we have a number of items, consider x_1 to x_n. Let each item x_j has a value p_j and a weight w_j associated to it. The maximum weight capacity of the bag is C.

The 0/1 knapsack problem has a particular constraint that we can either pick the item or leave it but we cannot break the item.

We can represent the objective and constraints of the 0/1-knapsack problem as follows:

$$\max \sum^{n} p_j.$$

given that

$$\sum^{n} w_j x_j \leq c, \qquad x_j = 0 \text{ or } 1,$$

The bounded knapsack problem puts limitation in the number of each item to be included in the collection.

Mathematical representation of the bounded knapsack problem is as follows:

$$\max \sum^{n} p_j.$$

given that

$$\sum^{n} w_j x_j \leq c, \qquad 0 \leq x_j \leq b_j,$$

The unbounded knapsack problem doesn't put any limitation on the number of each item selected.

There is another special case for Knapsack problem which is also a decision making problem where either we take the item or we leave it. The extra condition here is that the cost is equal to the value for every item in the list: C=V.

The special case of Knapsack problem is to find whether any subset of the given set of integers adds up to C exactly, or if the items are associated with negative costs and C is required to be zero, whether any subset adds up to exactly 0. This is known as the subset sum problem. In cryptography the problems are referred to as "knapsack problem" when what is actually meant is the "subset sum problem".

The Dynamic Programming solution of 0-1 Knapsack is an NP-complete problem and a pseudo polynomial time algorithm. The knapsack problem as well as the subset sum problem are considered to be NP-hard.[3] This has resulted in using of the subset sum for public key cryptography systems, like Merkle-Hellman. They use some groups other than integers. Binary Knapsack problem is considered to be NP-Complete.

A. Dynamic Programming Algorithm

Dynamic programming algorithm solves the knapsack problem in pseudo-polynomial time. In approaching the unbounded knapsack problem through dynamic programming we consider costs, let from c_1 to c_n and values from v_1 to v_n. Our aim is to select items in such a way that it maximizes the total value while the total cost is less than or equal to C.[2]

Then for each c[item] ≤ C, we define V(item) as the maximum value that can be obtained with total cost <= c[item]. Thus, V(C) is the solution to the problem.

We initialize the table having number of items+1 rows and C+1 columns with zeros. We then tabulate the results from V(0) to V(C) to get the optimal solution. Since the computation of each V(item) involves considering *'item'* number of items which have been already calculated, and their values filled in the table, and there are C values of V(item) to calculate, the time complexity through the dynamic programming approach thus belongs to the order of O(nC).

For achieving this we take a recursive function, V(item, cost) to be the maximum value that can be obtained with cost <= *'cost'* including items up to *'item'*.

We can define V(item,cost) recursively as follows:
V(0, cost) = 0
V(item, 0) = 0
V(item, cost) = V(item - 1, cost) if c[item] > cost V(item, cost) = max(V(item - 1, cost), v[item] + V(item - 1, cost – c[item]))
if c[item] ≤ cost

The computation of V(n, C) will give the final solution to the problem. Storing the previously calculated values in a table proves to be an efficient way. The time and space complexity of the solution is thus O(nC).

II. KNAPSACK FOR SHOPPING CART

In current ecommerce scenario, website navigation and product selection is a tedious task for users sensitive to price and time. [4][5] It was found that customers shop online because of conveniences such as time saving, ease of use, and reduced shopping stress among others. [8][9]

The algorithm for solving the 0/1 Knapsack problem can be implemented in e-commerce websites as well, for easy shopping experience for those users who have a fixed budget. The online customers can fix their budget before shopping, and fill the shopping cart by choosing items that are sorted as per the least price and/or the most relevance or rating. Items from the different categories that the shopper wants to buy from, can be clustered automatically by their popularity into baskets, wherefrom he can choose his best fit option. So the customer can get a number of items of value without exceeding the cost or budget.

So each item x_j in the e-commerce portal has a relevance p_j and a cost w_j. The maximum cost that we can put in the online cart is C, that is the fixed budget. This can be related with the classic knapsack problem where each item x_j has a value p_j and a weight w_j; and the items are to be put in a bag that can carry a maximum weight of C. Our goal through the algorithm is to maximize:

$$\sum^{n} p_j. \quad \text{subject to condition}$$

$$\sum^{n} w_j x_j \le c, \qquad x_j = 0 \text{ or } 1,$$

So, through this we try to maximize the relevance while minimizing the cost, so that more number of items can come within the budget. This would be a very easy to use technique for frequent and casual shoppers who have time, plan and funds/resources/budget constraint.

In current shopping sites checking the budget is manual, it has to be taken care by the customer himself again and again while choosing products from various categories. Consider he has to fill 10 products from 10 different categories in his cart. This would need him to check whether total price is exceeding his budget every time he adds an item, going back and forth in categories to change his selections, which would result in increase of time with every search.

If it is instead taken care by the website itself, it would guarantee maximum number of products at optimal price rates and at minimal time.

III. EXPERT SYSTEM FOR ONLINE SHOPPING CART MANAGEMENT

i. The User enters the Online Shopping Portal. He is asked to answer certain questions that is fed into the **Knowledge Base** through the **Knowledge Acquisition System** as Known Facts:
   - Total Budget
   - The Categories
ii. The system searches through the products database i.e. the **Knowledge Base** and sorts the items in the

chosen categories as per cost and provides three options to the user in form of selection buckets as per the following criteria:

- User ratings and reviews
- Number of items
- Expensive Branded Goods

iii. The **Inference Engine** contains the 0/1 Knapsack Algorithm along with the bounding conditions in form of rules that are applied on the above data.

- IF cost of item> Budget THEN
  Discard the item from cart
  ELSE
  Include the Item to cart
- IF Total cost of the items in cart > Total Budget THEN
  Discard the item put in the cart last
  ELSE IF Total cost of the items in cart< Total Budget THEN
      IF Cost of next Item in Items List<=Remaining Budget THEN
      Include the Item In Cart
      ELSE
      Discard the Item
  ELSE
  Consider the Cart Full and Display The Final Items
- The relevance of items should be maximum i.e user ratings or number of items or expensive branded goods included in the final items list, as per the user's choice.

iv. The User selects one of the buckets on which the Inference Engine starts working with an aim to fulfil the user requirement of total budget and categories of items given as input. It applies the Inference Rules to reach the appropriate answers.

Algorithm for Inference Engine:
Initialize the Knapsack capacity that is the total budget or cost.
Fetch the cost of all the items in a sorted order.
Fetch the relevance or rating of all the items.

```
N= number of items that we have in our selection basket.
V= empty 2D array of N+1 rows and budget+1 columns
for i from 0 to N+1 do:
        for j from 0 to budget+1 do:
                if i = 0 or j =0 then:
                        V[i,j]:=0
                else:
                        if cost[i-1] > j then:
                        V[i,j] := V[i-1, j]
                        else:
                        V[i,j] := max(V[i-1, j], V[i-1,j-cost[i-1]]+ relevance[i-1])
items=[]
while(N!=0 and budget!=0):
        if(V[N,budget] in V[N-1]):
                N=N-1
                continue
        Else:
                Include N in items
                N=N-1
                budget=budget-cost[N]
return items
```

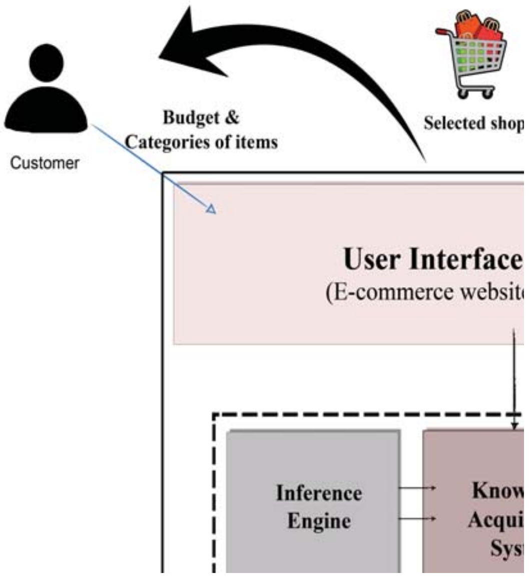
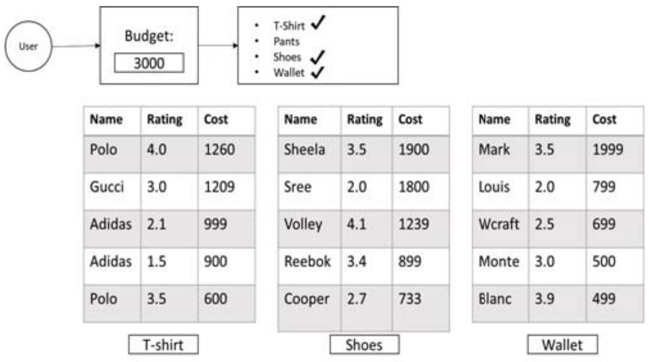


Fig. 2. Architecture of procedure based expert system for online shopping cart management

IV. IMPLEMENTATION

Suppose a user has to buy a few products from an e-commerce website and his budget is 3000/-. The user needs to select certain categories, let's say T-Shirt, Shoes and Wallet. The items are already sorted in accordance to their cost price in the database as shown in the Fig.2.



Name	Rating	Cost
Polo	4.0	1260
Gucci	3.0	1209
Adidas	2.1	999
Adidas	1.5	900
Polo	3.5	600

T-shirt

Name	Rating	Cost
Sheela	3.5	1900
Sree	2.0	1800
Volley	4.1	1239
Reebok	3.4	899
Cooper	2.7	733

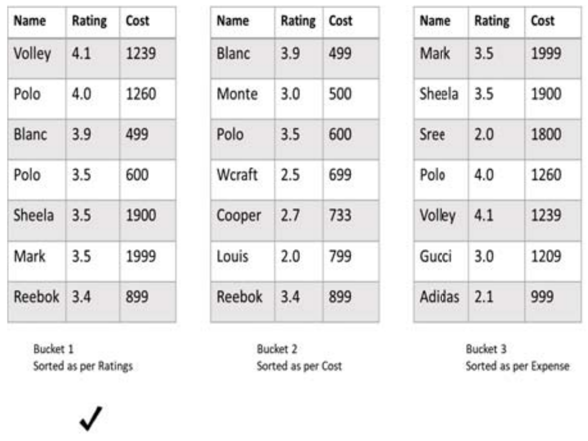
Shoes

Name	Rating	Cost
Mark	3.5	1999
Louis	2.0	799
Wcraft	2.5	699
Monte	3.0	500
Blanc	3.9	499

Wallet

Fig. 3. Customer Requirement and Knowledge Acquisition from Knowledge Base

When the user selects the categories of products he has to buy, i.e. T-Shirt, Shoes and Wallet, the expert system prepares three separate selection buckets as shown in Fig.3. In the first bucket the items are sorted as per the ratings from the three categories. In the second bucket the sorting is as per the minimum cost and in the third selection bucket the sorting is as per the highest price or expense.



Name	Rating	Cost
Volley	4.1	1239
Polo	4.0	1260
Blanc	3.9	499
Polo	3.5	600
Sheela	3.5	1900
Mark	3.5	1999
Reebok	3.4	899

Bucket 1
Sorted as per Ratings

✓

Name	Rating	Cost
Blanc	3.9	499
Monte	3.0	500
Polo	3.5	600
Wcraft	2.5	699
Cooper	2.7	733
Louis	2.0	799
Reebok	3.4	899

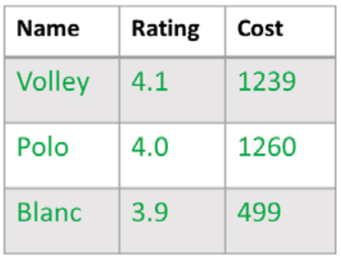
Bucket 2
Sorted as per Cost

Name	Rating	Cost
Mark	3.5	1999
Sheela	3.5	1900
Sree	2.0	1800
Polo	4.0	1260
Volley	4.1	1239
Gucci	3.0	1209
Adidas	2.1	999

Bucket 3
Sorted as per Expense

Fig. 4. Selection Buckets Provided by The Expert System

The user selects one bucket as per his priority. Here Bucket 1 is selected as shown in Fig.3. The Inference Engine then takes the list of items in Bucket 1 along with their cost and rating and processes it to provide the optimized selection of items for the shopping cart.



Name	Rating	Cost
Volley	4.1	1239
Polo	4.0	1260
Blanc	3.9	499

Fig. 5. Optimized Shopping Cart Prepared by Inference Engine

V. CONCLUSION

Using the case it can be concluded that the Shopping Cart Management as a 0/1 Knapsack Problem can be better solved through an Expert System which is following Dynamic Programming approach. The proposed system provides an opportunity to the customers to see different cart on different criteria, also would provide ease of use to the customers for online shopping. Hence, such a solution can be overall a happy and good experience for a customer. The same knowledge base can be utilized to provide the rules, procedures and data for another kind of problem in similar kind of domain specific problem.

REFERENCES

[1] He, Y. C., Wang, X. Z., He, Y. L., Zhao, S. L., & Li, W. B. (2016). Exact and approximate algorithms for discounted {0-1} knapsack problem. *Information Sciences*, *369*, 634-647.

[2] Lv, J., Wang, X., Huang, M., Cheng, H., & Li, F. (2016). Solving 0-1 knapsack problem by greedy degree and expectation efficiency. *Applied Soft Computing*, *41*, 94-103.

[3] König, D., Lohrey, M., & Zetzsche, G. (2016). Knapsack and subset sum problems in nilpotent, polycyclic, and co-context-free groups. *Algebra and Computer Science*, *677*, 138-153.

[4] Natarajan, T., Balasubramanian, S. A., & Kasilingam, D. L. (2017). Understanding the intention to use mobile shopping applications and its influence on price sensitivity. Journal of Retailing and Consumer Services, 37, 8-22.

[5] Shanbhag, S., Nair, S., Nai, N., & Shaikh, B. (2016). Intelligent Shopping Agent.

[6] Ingham, J., Cadieux, J., & Berrada, A. M. (2015). e-Shopping acceptance: A qualitative and meta-analytic review. Information & Management, 52(1), 44-60.

[7] Yang, K., Li, X., Kim, H., & Kim, Y. H. (2015). Social shopping website quality attributes increasing consumer participation, positive eWOM, and co-shopping: The reciprocating role of participation. Journal of Retailing and Consumer Services, 24, 1-9.

[8] Hasan, B. (2016). Perceived irritation in online shopping: The impact of website design characteristics. Computers in Human Behavior, 54, 224-230.

[9] Al Karim, R. (2013). Customer Satisfaction in Online Shopping: a study into the reasons for motivations and inhibitions. IOSR Journal of Business and Management, 11(6), 13-20.

[10] Feng, Y., Wang, G. G., Deb, S., Lu, M., & Zhao, X. J. (2017). Solving 0–1 knapsack problem by a novel binary monarch butterfly optimization. *Neural computing and applications*, *28*(7), 1619-1634.

[11] D. Bhargava and S. Vyas, "Agent based solution for dining philosophers problem," 2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS), Dubai, 2017, pp. 563-567. doi: 10.1109/ICTUS.2017.8286072

[12] Vyas, V., Saxena, S., & Bhargava, D. (2015). Mind Reading by Face Recognition Using Security Enhancement Model. In Proceedings of Fourth International Conference on Soft Computing for Problem Solving (pp. 173-180). Springer India