www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 3 Issue 6 June, 2014 Page No. 6526-6530

Determining K-most demanding products using data mining technique

***Sonal K. Bang*[1]*, Prof. P. N. Kalavadekar*[2]**

[1]PG student, Depertment of Computer Engineering,
SRES's College of engg,University of Pune, Kopergaon-423603, India
sonalbang@rediffmail.com

[2]Assot. Professor, Depertment of Computer Engineering,
SRES's College of engg, University of Pune, Kopergaon-423603, India,
kprak3004@gmail.com

Abstract*: This paper formulates a problem for production plan as k- most demanding products (k-MDP). Given a set of customers demanding a certain type of products with multiple attributes, a set of existing products of the type, a set of candidate products that company is able to offer, and a positive integer k, it helps the company to select k products from the candidate products such that the expected number of the total customers for the k products is maximized. One greedy algorithm is used to find approximate solution for the problem. Attempt is also made to find the optimal solution of the problem by estimating the expected number for the total customers of a set of k candidate products for reducing the search space of the optimal solution.*

Keywords: Algorithm for knowledge management, data mining, decision support, performance evaluation of algorithm**.**

1. Introduction

Microeconomics deals with the customers and producer relationship and how they take decisions. The major concern of microeconomics is the customer preferences which is an important factor in making decisions related to product sales. While making production plans or marketing strategies related to the product, company needs to identify the product with maximum value or high utility (Chen 1999). This utility or values can be treated as the function of interaction of the company with the other entities such as customers and competitors. Taking competition into account, ht e problem considered in this paper, is to find the highest utility production plans for the company where the utility of the production plan is decided is decided by the expected number of total customers for the product in plan.

The problem can be defined as, discovering k- most demanding products (k-MDP), given set of customers demanding specific type of product with some attribute, set of existing product with same attribute, set of candidate product that a company can offer, then we can help company to find k candidate products such that for these k product the expected number of total customers is maximized.

2. Related Work

Kleinberg et al.[5] pointed out that data mining technique can be used to solve the several problems related to
Microeconomics. Microeconomics related problems can be divided into three different categories as potential customer finding, product advantage discovery, and product positioning. The problem of potential customers finding has been tried to solve in paper [2,5,7,9,12,13,16].

Reverse k-nearest neighbor query is used in paper [2,3,5,6]:

In k-nearest neighbor algorithm, we try to find all objects in data set with k nearest neighbor as per specification of query object. In this paper an efficient algorithm has been presented to find out k-nearest neighbor.

The method called ***reverse skyline query*** is used in paper [7,8]: In reverse skyline queries, multidimensional dataset P is considered to a query point q. Here in data space point q becomes origin & all point of P are represented by their distance vector to q. The reverse skyline query algorithm the objects whose dynamic skyline contain query object q.

In all these research papers input is customer preferences & product name & output is customers whose favorite products contain the specified product all to their preferences.

Development of a new technique to find products which are popular with customers by determining right positioning as per production plan is given in papers. Drawback of this method is although they consider production plan. They do not consider customer preferences. The problem of finding top customers for one specific product has been solved in papers [9,10]. Drawback is it considers only one product. They do not consider customer preferences.

The problem of finding top customers for one specific product has been solved in paper [3]. Drawback is it consider only one product.

In this paper both product competition and customer requirements are taken into consideration to find k products from the candidate products that a company can offer so as to maximize the expected number of the total customers for the k products.

3. K-MDP Problem

Suppose that there is a set of customers C, demanding particular product, EP is the set of existing product, d is the attributes describing quality of product. The quality of product ep in EP is represented as vector {ep1,ep2,...ep[d]}, where ep[i] denotes the value of ep for ith quality describer. In addition, for a customer c in C, the requirements on the products are represented by a vector {c[1], c{2], . . . c[d]},describing the quality constraint on the value of each quality describer. Suppose that there is a nonempty set of candidate products, denoted CP, this can be offered by a company. The quality of a candidate product is also represented by a vector of the quality describers.

Let kCP denote a set of k products chosen from CP, cp denote a product in kCP, and c denote a customer in C. In addition, N(EP,c) and N(kCP, c) denote the total number of products in EP and kCP satisfying c, respectively. As c will definitely purchase one of the products satisfying his/ her requirements, if cp satisfies the requirements of c, the probability of c purchasing cp is inverse proportional to the total number of products in EP and kCP satisfying c, otherwise, it is 0. Consequently, the probability of a customer c purchasing a product cp in kCP, denoted P(cp, c) is calculated as follows:

$$P(cp,c) = \frac{1}{\mathrm{N(EP,c)} + \mathrm{N(kCP,c)}}$$

$$= 0, \text{otherwise} \quad (1)$$

The expected number of the total customers in C for the set kCP, denoted E(kCP,C) is defined by adding the expected number of the customers in C for each product cp in kCP as follows:

$$E(kCP,C) = \sum_{cp \in kcp} \sum_{c \in C} p(cp,c) \quad (2)$$

Definition of k-MDP: Given a set C of customers, a set EP of existing products, a set CP of candidate products, and a positive integer k which is less than number of CP, the k-MDP are the k products chosen from the set CP with the maximum E(kCP,C).

4. Algorithms

This section gives an idea about how the bitmap structure is used to maintain the satisfying information of the data in EP& C. Then 2 greedy algorithms are proposed to solve the k-MDP discovering problem.

4.1 BMI Index Structure

For the pre-processing of the data collected a bitmap index is generated taking the attributes of the customers into consideration. These attribute values are inserted by customers. Bitmap Index is collection of customer preferences in binary format.

Bitmap is completely non-blocking and exploits a bitmap structure to quickly identify whether a point is an interesting point or not. Each record is mapped into a m-bit vector, where m is the sum of the number of distinct attribute values over all dimensions. Unlike existing bitmap structures which are typically a bitmap version of the entire database, our bitmap structure is a pre computed bit structure with more information.

4.2 Greedy algorithms

4.2.1 Single product based Greedy algorithm:

Let S denote a set containing a single candidate product cp in CP. The SPG algorithm uses E(S, C), which computes the expected number of the customers in C for S, as the ranking function of the candidate products. The candidate products with the top-k values of the ranking function are selected to form an approximate solution of the k-MDP discovering problem.

Algorithm 1: The SPG Algorithm

Input: N_vector(EP,C), set C of customer requirements, set CP of candidate product, and the value k

Output: A set of k candidate product

```
1.    i=0;
2.    For each candidate product cp in CP
3.    {compute the satisfaction bit string of cp;
4.       S={cp};
5.       Compute E(S,C);
6.       i=i+1;
7.      If i<=k
8.         Insert E(S,C)  into the top-k list
9.     Else if E(S,C)> the smallest value in top-k list
10.   Replace the smallest in top-k list with E(S,C);}
11.   Put the corresponding candidate products in the top-k
      list to set the kCP;
      Return kCP;
```

4.2.2. Incremental base Greedy algorithm:

Let S denotes an empty set initially. In each iteration, the IG algorithm selects one of the unselected candidate products cp, which has the maximum E(S ∪ {cp}, C) value. After inserting the selected candidate product into S, the values of the selection function for the unselected candidate products are recomputed in the following iteration to decide the next selected candidate product. The above process continues until k candidate products have been selected.

Algorithm 2: The IG algorithm

Input: N_vector(EP,C), set C of customer requirements, set CP of candidate product, and the value k

Output: A set of k candidate product

```
1.    For each candidate product cp in CP
2.    Compute the satisfaction bot string of cp;
3.    S= null;
4.    While /S/<k
5.    {  max_E=0;
6.       For each candidate product cp in CP
7.       {  temp_S=S ∪ {cp};
8.            Compute  E(temp_S,C);
9.            If E(temp_S,C)>max_E
10.           {     max_P={cp};
11.                 Max_E= E(temp_S,C)  }}
12.      S=S ∪ max_P;
13.      CP=CP – max_P;   }
14.   Return  kCP;
```

4.3 Optimal Algorithms

4.3.1Apriori based algorithm:

Similar to the Apriori algorithm, the APR algorithm generates all the sets containing a single candidate product first. Let S denote a set of l candidate products, where 1 < l < k. For any kCP which contains S, denoted kCP_S, the main idea of the APR algorithm is to estimate the upper and lower bounds of E($kCP_{S,}$ C). The bound values are used to prune the sets of l candidate products whose supersets are impossible becoming the optimal solution of the k-MDP discovering problem. In the next iteration, the remaining sets of l candidate products ($1 < l < k$) are combined to generate the sets of (l+1) candidate products. The above process will repeat until the sets of k candidate products are generated to discover the k-MDP.

$$UBE(kCPs, C = \sum_{c\in C} UBE(kCPs,\{c\}) \quad (3)$$

$$LBE(kCPs, C) = \sum_{c\in C} LBE(kCPs,\{c\} \quad (4)$$

Algorithm 3: The ARP algorithm

Input: N_vector(EP,C), set C of customer requirements, set CP of candidate product, and the value k

Output: A set of k candidate product

```
1.   For each candidate product cp in CP
2.        Compute the satisfaction bit string of cp;
3.   $CPS_1$=null;
4.   For each candidate product cp in CP
5.        $CPS_1 = CPS_1 \cup \{\{cp\}\}$;
6.   For (l=1; l<k; l++)
7.   {  MAX=0;
8.        For each S in CPS1
9.        {   Compute UB_E(kCPs,C);
10.            Compute LB_E(kCPs,C);
11.            If(LB_E(kCPs,C)>MAX)
               MAX+LB_E(kCPs,C);}
12.       $CPS_l$ = Candidatecheck( MAX,$CPS_l$);
13.       $CPS_{l}+_1$ = Apriori_Gen($CPS_l$);  }
14.   MAX=0;
15.  For each S in $CPS_k$
16.  {   Compute E(S,C);
17.         If (E(S,C)>MAX)
18.            kCP=S;  }}
19.   Return kCP;
```

Function CandidateCheck(*MAX, CPS*$_l$)

```
1.   For each S in $CPS_l$
2.     If ( MAX > UB_E(kCPs,C))
3.        $CPS_l = CPS_l - \{S\}$;
4.   Return $CPS_l$;
```

Function Apriori_Gen(*CPS*$_l$)

```
1.   $CPS_{l+1} = CPS_l \oplus CPS_l$;
2.   For each S in $CPS_l$
3.      If(( any subset of S with size l) $\notin CPS_l$)  $CPS_{l+1}$=
        $CPS_{l+1} - \{S\}$;
4.   Return $CPS_{l+1}$;
```

4.3.2. Upper bound pruning Algorithm:

The solution found by SPG algorithm approaches the optimal solution. Therefore, the UBP algorithm takes the solution found by the SPG algorithm as the baseline solution kCP_b. Besides, for each set kCP of k candidate products, another method is proposed to efficiently estimate the upper bound of E(kCP,C). The value of E(kCP,C) is required to be computed only when the upper bound of E(kCP,C) is larger than E(kCP_b, C). The baseline solution kCP_b will be updated if a better solution is found. Therefore, the final result of kCP_b is the optimal solution. Moreover, a well-designed method to decide the checking order of the sets of k candidate products is provided such that the checking process of the UBP algorithm can be performed efficiently.

Algorithm 4: The UBP Algorithm

Input: N_vector(EP,C), set C of customer requirements, set CP of candidate product, and the value k

Output: A set of k candidate product

```
1.   For each candidate product cp in CP
2.   {  compute the satisfaction bit string of cp;
3.        S={cp};
4.        Compute E(S,C);  }
5.   $SL=<cp_1', cp_2', ....cp_{|cp|}'>$ // according to decreasing values
     of E(S,C)'
6.   $kCP_b=\{cp_1', cp_2', ......cp_k'\}$;
7.   base=E($kCP_b$,C);
8.   $kCP=\{cp_1', cp_2', .....cp_{k-1}', cp_{k+1}'\}$;
9.   $prune=\{cp_{|cp|-k+1}', cp_{|cp|-k+2}', ....,cp_{|cp|}\}$;
10.  While (true)
11.  {   compute UB_E2(kCP,C);
12.       If UB_E2(kCP,C)>base
13.       {   compute E(kCP,C);
14.             If E(kCP,C)>base
15.             { base=E(kCP,C);
16.                   $kCP_b$=kCP;   }}
17.       Else prune=kCP;
18.       kCP=NextCandidateGen(SL,prune,k);
19.       If  kCP= null   Break;  }
20.     Return $kCP_b$;
```

Fuction NextCandidateGen(*SL,prune,k*)

```
1.   While(true)
2.   {kCP= the next set of candidate product according to t
3.        If prune $<_r$ kCP   continue;
4.        Else  break;  }
5.    Return kCP;
```

5. Performance Evaluation

To evaluate efficiency & accuracy the algorithms were implemented in java with SQL database.

The Synthetic data is considered by creating an online shopping site for electronic products where the data collected.

5 Synthetic test databases were considered for performance evaluation as given in Table 5.1.

Table 5.1: Test Databases

Sr. No.	Database Name	Description
1	DB1	150 customers & 10 products
2	DB2	250 customers & 10 products
3	DB3	250 customers with number of customers changed per product
4	DB4	250 customers with changed value of product attribute
5	DB5	250 customers with 3 product attributes

5.1. Finding accuracy:

Table 5.2 shows the accuracy of each algorithm in percentage on different databases. Fig 5.1 shows the graph for percentage accuracy of the algorithms.

Table 5.2: Accuracy of each algorithm

	DB1	DB2	DB3	DB4	DB5
SPG	100%	66%	100%	33%	100%
IG	100%	66%	100%	66%	100%
ARP	33%	33%	33%	33%	33%
UBP	100%	100%	100%	100%	66%

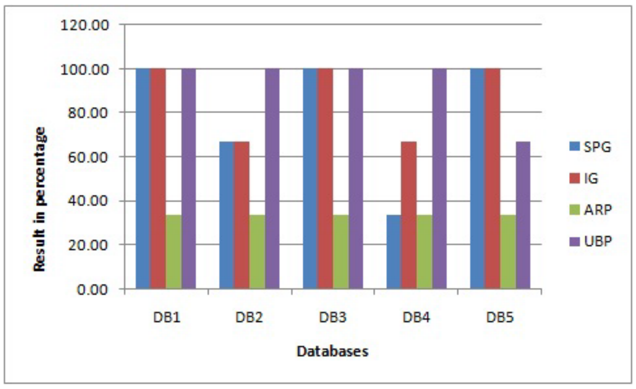

Figure 5.1: Bar chart for accuracy of algorithms

5.2. Finding time efficiency:

Table 5.3 shows the time (in sec.) required for each algorithms on different databases. Fig. 5.2 shows the graph for the time require for algorithms to execute.

Table 5.3: Result of time efficiency (in sec.)

	DB1	DB2	DB3	DB4	DB5
SPG	1.178	1.201	1.001	1.318	2.987
IG	2.102	2.154	1.746	2.322	2.428
ARP	0.003	0.004	0.002	0.001	0.002
UBP	0.847	0.909	0.648	0.737	0.678

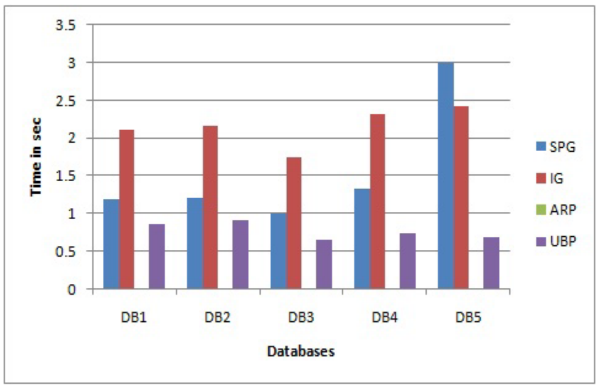

Figure 5.2: Bar chart for time required for algorithms

5.3. Finding database effect:

For finding database effect we have increased the number of customers for 3 attribute products tested the time required for execution of algorithms. Table 5.4 shows the time of each algorithm on increasing transactions. Fig. 5.3 shows the graph for same.

Table 5.4: Result of database effect (in sec.)

	250	*350*	*500*	*700*
SPG	2.587	2.779	3.249	3.549
IG	2.428	2.633	2.856	3.036
ARP	0.002	0.001	0.002	0.001
UBP	0.678	0.645	0.784	0.942

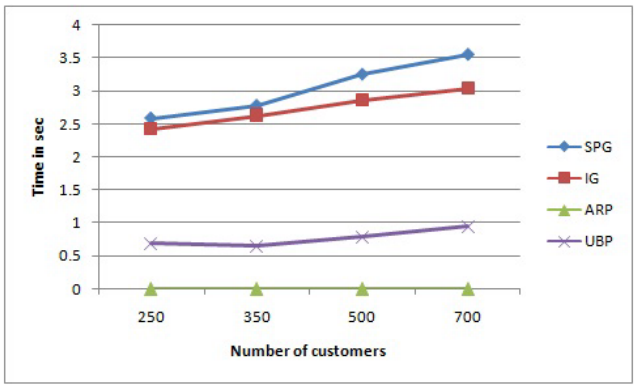

Figure 5.3: Line chart for database effect

5.4. Analysis & discussion:

In this section we have analyzed the algorithms based on the experimental setup made. From table 5.3 and table 5.2 we can observe that the time required for ARP algorithm is minimum but the accuracy of the ARP algorithm is less. Whereas the accuracy of the UBP algorithm is more than any other algorithm & also the time required for this algorithm is minimum. So we can conclude that UBP algorithm is better than any other algorithm.

Also from table 5.4 we have observed that the all the algorithms execute properly for more than three attributes of the product. The time required for SPG & IG algorithms go on increasing as the database size increases where as the time for ARP & UBP algorithm remains constant. And UBP algorithm is better for more than 2 attributes.

6. Conclusion

In this paper k-most demanding product system is implemented. It consist of four stages in first stage the data is collected from the registration of the customers, in second stage preprocessing is done on this data to generate bitmap index, In next stage 4 different algorithms are implemented to find the top demanding products. At the end the algorithms are analyzed to find the better algorithm.

We have created an online electronic shopping site for synthetic dataset. The databases of about 250 customers & 10 products have been considered. At the start the system was implemented for products with 2 attributes after the successful execution of this system, products with 3 attributes is considered.

The experimental results show that the system is working for more than 2 attributes of the product. The UBP has accuracy of 100% & time efficiency as 0.678 sec. The ARP has accuracy of 33% & time efficiency as 0.001 sec. So we can say that UBP algorithm is better than any other algorithm.

References

[1] Chen-Yi Lin, Jia-Ling Koh, and Arbee L.P. Chen, ”Determining k-Most Demanding Products with Maximum Expected Number of Total Customers”, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING.Harlow, England: Addison-Wesley, 2013.

[2] J. Kleinberg, C. Papadimitriou, and P. Raghavan, ”A Microeconomic View of Data Mining”,Data Mining and Knowledge Discovery, vol. 2, no. 4, pp. 311-322, 1998.

[3] E. Achtert, C. Bohm, P. Kroger, P. Kunath, A. Pryakhin, and M. Renz, ”Efficient Reverse k-Nearest Neighbor Search in Arbitrary Metric Spaces”,Proc. 25th ACM SIGMOD Intl Conf. Management of Data, pp. 515-526, 2006.

[4] S. Borzsonyi, D. Kossmann, and K. Stocker, ”The Skyline Operator”, Proc. 17th Intl Conf. Data Eng., pp. 421-430, 2001.

[5] Y. Tao, D. Papadias, and X. Lian,”Reverse kNN Search in Arbitrary Dimensionality”, Proc. 30th Intl Conf. Very Large Data Bases, pp. 744- 755, 2004.

[6] W. Wu, F. Yang, C.Y. Chan, and K.L. Tan,”FINCH: Evaluating Reverse k-Nearest-Neighbor Queries on Location Data”, Proc. 34th Intl Conf. Very Large Data Bases, pp. 1056-1067, 2008.

[7] E. Dellis and B. Seeger, and K.L. Tan,”Efficient Computation of Reverse Skyline Queries”, Proc. 33rd Intl Conf. Very Large Data Bases, pp. 291- 302, 2007.

[8] X. Lian and L. Chen, and K.L. Tan,”Monochromatic and Bichromatic Reverse Skyline Search over Uncertain Databases”,Proc. 27th ACM SIGMOD Intl Conf. Management of Data, pp. 213-226, 2008.

[9] C. Li, B.C. Ooi, A.K.H. Tung, and S. Wang,”DADA: A Data Cube for Dominant Relationship Analysis”, Proc. 25th ACM SIGMOD Intl Conf. Management of Data, pp. 659-670, 2006.

[10] Q. Wan, R.C.-W. Wong, I.F. Ilyas, M.T. Ozsu, and Y. Peng,”Creating Competitive Products”, Proc. 35th Intl Conf. Very Large Data Bases, pp. 898-909, 2009.

Author Profile

Miss. Sonal Kamalkishor Bang M.E-II, Computer Engineering, SRES COE, Kopargaon, University of Pune, Maharashtra, India. sonalbang@rediffmail.com.

Prof P. N. Kalavadekar Associate-Professor, Computer Engineering, SRES COE, Kopargaon, University of Pune, Maharashtra, India. kprak3004@gmail.com