International Journal of Recent Trends in Engineering & Research
ISSN (ONLINE) : 2455 – 1457
IMPACT FACTOR : 4.101

ICNSCET19- International Conference on New Scientific Creations in Engineering and Technology

Multi-Constraint Optimal Skyline Product Combination Under Price Promotion Using TLE Algorithm.

Mr. K.Vignesh saravanan*, Ms. J Bharathy, Dr. K.Vijayalakshmi*****
**Assistant Professor, CSE Department, Ramco institute of technology*
***UG Student, CSE Department, Ramco institute of* technology
****Professor and Head, CSE Department, Ramco institute of technology*

Abstract— Nowadays, with the development of e-commerce, a growing number of customers choose to go shopping online. To find out attractive products from online shopping marketplaces, the skyline query is a useful tool which offers more interesting and preferable choices for customers. The skyline query and its variants have been extensively investigated. However, to the best of our knowledge, they have not taken into account the requirements of customers in certain practical application scenarios. Recently, online shopping marketplaces usually hold some price promotion campaigns to attract customers and increase their purchase intention. Considering the requirements of customers in this practical application scenario, we are concerned about product selection under price promotion. We formulate a constrained optimal product combination (COPC) problem. It aims to find out the skyline product combinations which both meet a customer's willingness to pay and bring the maximum discount rate. The COPC problem is significant to offer powerful decision support for customers under price promotion, which is certified by a customer study. To process the COPC problem effectively, we first propose a two list exact (TLE) algorithm. The COPC problem is proven to be NP-hard, and the TLE algorithm is not scalable because it needs to process an exponential number of product combinations. Additionally, we design a lower bound approximate (LBA) algorithm that has guarantee about the accuracy of the results and an incremental greedy (IG) algorithm that has good performance.

.**Keywords**— Data management, price promotion, skyline query, NP-hard.

INTRODUCTION

With the development of e-commerce, a growing number of customers choose to go shopping online because it saves time and effort. However, it always contraries to expectations of customers. This is because they may need to pick up one choice among thousands of products. To help customers identify attractive products, a skyline query is admittedly a common and effective methodology. According to the definition of the skyline query [1], a product which is not dominated by any other product is said to be a skyline product or it is in the skyline. The products in the skyline are the best possible tradeoffs between all the factors that customers care about. The skyline query is useful in identifying attractive products. In Dingdongs and Alibaba's Taobao Mall which are the most famous online shopping malls in China, there are many online stores that specialize in one category of products such as red wine, watches, television, laptop, to name just a few. During the weekends or holidays, these stores usually hold some price promotion campaigns to boost consumption. Under the price promotion campaigns of these stores, a customer could select an optimal product combination by himself. Besides, the customer is common to participate in cooperation with his families or friends for group-buying.

II. RELATED WORKS

As an important data management operator, the skyline query and its variants has received a great attention in the literature . In our COPC problem, itcomputes the optimal skyline product combinations with a constraint, which is the customer's willingness to pay. The closely related problems are group skyline queries and skyline queries under constraints, and the related works are reviewed in this section.

2.1 GROUP SKYLINE QUERIES

The skyline query aims to return the points that are not dominated by any other point . However, most of the works about the skyline query just analyze individual points, and they are inappropriate to many applications that call for analysis of groups of different points. Motivated by this, group skyline queries are developed and paid growing attention.

In most of the group skyline queries, optimal groups are computed by the dominance relationship between corresponding aggregate-based points of different groups formulated top k combinatorial skyline query (k-CSQ). It returns those combinatorial skyline tuples whose aggregate values for a certain attribute are maximum. Since only the first k combinations are required, the k-CSQ query process can be simplified extended the traditional skyline queries and formulated a combinatorial skyline query, namely CSQ, which is to find the outstanding skyline combinations studied the group skyline query which is based on the dominance relationship between the groups of the same size. The dominance relationship is checked according to the aggregate values of attributes. introduced aggregate skylines, where the skyline works as a filtering predicate on sets of records. The aggregate skyline queries merge the functionalities of

two basic database operators, skyline and group by focused on a novel problem of groups of *k* tuples, which are not dominated by any other group of equal size, based on aggregate-based group dominance relationship. They also identified two anti-monotonic properties to filter out candidate groups researched the work which is similar to the group skyline computation. They focused on a problem of creating competitive products which are not dominated by the products in the existing market. Here each new product is generated by combing products from different source tables. To reduce the search space, it only combines the skyline products from source tables to generate new products. In addition, they also presented an approach which divides similar products into groups and processes them as a whole.

2.2 SKYLINE QUERIES UNDER CONSTRAINTS

Skyline query is a useful tool to find out attractive products which offer more interesting and preferable choices for customers. However, the size of the skyline query results cannot be controlled flexibly. Accordingly, many research efforts have been devoted to contend with this problem. The existing approaches to address this problem are developed to identify *k* representative skylines which have the maximum dominant capacity or the maximum diversification were concerned about the case when the actual cardinality of skyline results is less than the desired result cardinality *k*.

They proposed a new approach, namely skyline ordering, which forms a skyline-based partitioning of a given dataset. Then they applied a set-wide maximization technique, which is used to find an object set dominating the largest number of points, to process each partition focused on how to choose *k* representative skylines over data streams formulated the most desirable skyline object query which reports the most preferable *k* skylines based on a new ranking criterion studied the problem of selecting *k* skyline points such that the number of points, which are dominated by at least one of these *k* skyline points, is maximized proposed the *K*-dominating query which retrieves *K* points dominating the largest number of other points. This query does not necessarily contain skyline points but has the advantages of both ranking queries and skyline queries, which are with the control on the size of the answer set and without users' efforts to specify ranking functions formulated a top *k* ranking problem which retrieves points appearing frequently in the subspace skylines. Identified the problem of finding top-*k* profitable products. Given a set of packages in the existing market and a set of potential new packages, they wanted to select *k* new packages such that the sum of the profits of the selected packages is maximized and each selected package is not dominated by any package in the existing market and any selected new package. Lin et al. [13] proposed a *k*-most demanding products (k-MDP) discovering problem that helps the company to select *k* products from the candidate products with the maximum expected number of the total customers. In [20], we developed a top *k* favourite probabilistic product query. It is

utilized to select k products which can meet the needs of a customer set at the maximum level.

Notation	Definition
P	the product dataset
N	the size of the product dataset
SP	the skyline product set over P
N_S	the size of SP
SP'	the skyline product combination with $SP' \subseteq SP$
WTP	a customer's willingness to pay
β, α	the price promotion campaign is getting $\$\beta$ off each $\$\alpha$ purchase
$MaxSize$	the maximum size of SP'
$MaxDisNum$	the maximum discount number
$MaxDisRate$	the maximum discount rate
OriPri(p)	the original price of the product p
Discount(p)	the discount by purchasing the product p
ActPay(p)	the actual payment of the product p
DisRate(p)	the discount rate of the product p

TABLE 1: The summary of frequently used notations.

III. THE CONSTRAINED OPTIMAL PRODUCT COMBINATION (COPC) PROBLEM

In the COPC problem, it needs to compute the skyline products by the skyline query which a useful tool for decision support is. The skyline query over all the attributes may give rise to loose some important product combinations. Assume that there are three products p1, p2, and p3 whose prices are \$190, \$210, and \$200, respectively, the other attributes of p2 and p3 are the same, and the price promotion campaign is "get \$60 off every \$200 purchase". In the skyline query over all the attributes, p2 is dominated by p1 and pruned since p2 has a lower price and the other attributes of them are the same. However, the discount rates of {p1, p2} and {p1, p3} are equal to 0.300 and 0.154 separately, and {p1, p2} is obvious a great choice with the maximum discount rate. For p2 is not in the skyline, the important product combination {p1, p2} is overlooked in the process of product selection. In [8], Liu et al. formulated a new G-Skyline query that aims to return optimal point groups, namely G-Skylines. Different from other group skyline queries, it reports more comprehensive results and may return optimal point groups (G-Skylines) that contain non-skyline points [8]. In essence, for a G-Skyline G, each non-skyline point p∈G is only dominated by some other point's p ′∈G.

Definition 1. (Skyline Query [1]): The skyline query returns all the products p∈P such that there does not exist any other product p ′∈P−p satisfying p ′≺p. In Section 1, we classify the present price promotion campaigns into two categories which are independent product and dependent-product selections. In this paper, we focus on the dependent-product selection, which includes the campaigns such as "get $β off every $α purchase" and "$β coupon every $α purchase" etc. This is because these campaigns are widely adopted by online shopping malls and much more complicated than the ones of the independent-product selection campaigns. In addition, under the campaigns of independent-product selection, the skyline query could offer powerful decision support. But, the skyline query only does little help when selecting products under the campaigns of dependent-product selection. In the following, we first research our problem under the price promotion campaign as "get $β off every $α purchase". In particular, by investigating from Jingdong and Alibaba's Taobao Mall, the two most famous online shopping malls in China, β and α are usually set to two and three-digit numbers, respectively. The popular price promotion campaign is getting β ′×10 off every α ′×100 purchase where β ′ and α ′ are integers and β ′ , α′∈[1, 9].

Definition 2. (Original Price): Given a product combination P ′ , its original price is computed as

$$\mathbf{OriPri(P')= \sum_{p \in P'} OriPri(p),}$$

where OriPri(p) is original price of a product p∈P ′

Definition 3. (Actual Payment): Given a price promotion campaign "get $β off every $α purchase ", the actual payment of a product p∈P is

$$\mathbf{ActPay(p)=OriPri(p)- \lfloor OriPri(p)\ \alpha \rfloor \times \beta.}$$

Definition 4. (Discount Rate): Suppose that the price promotion campaign is getting $β off every $α purchase. The discount rate gaining by selecting a product combination

$$\boldsymbol{DisRate(P)=[OriPri(P)\_\times\_OriPri(P')]}$$

:

Definition 5. (The Constrained Optimal Product Combination (COPC) problem): Given a set of products *P*, a customer's willingness to pay *WTP*, the COPC problem is to find the skyline product combinations *SP′*, such that they bring the maximum discount rate without exceeding the customer's payment willingness. The COPC problem can be formulated as

$$\boldsymbol{maximize\ DisRate(SP')}$$
$$\boldsymbol{subject\ to\ ActPay(SP') \leq WTP\ \ for\ SP' <= SP.}$$

Here *SP* denotes the skyline product set over *P*.

IV.ALGORITHMS FOR THE COPC PROBLEM

To process the COPC problem, a naive exact algorithm is to generate all the skyline product combinations which are not beyond the customer's payment willingness, compute the discount rate of each candidate combination, and identify the ones that bring the maximum discount rate.

Consider each combination of the skyline products within *SP*, which contains *t* products for $1 \leq t \leq MaxSize$. The (number of these combinations that contain *t* products is *NSt*), where *NS* represents the cardinality of the skyline set *SP*, *MaxSize* denotes the maximum size of the skyline product combinations.

Algorithm1:TLE Algorithm

Input: The skyline product set SP, a price promotion campaign "get $β off every $α purchase", and a customer's payment willingness WTP
Output: A result set SP∗ of the COPC problem
Algorithm Steps:

1. Divide SP into two parts: SP1={sp1 ,sp2 , . . . ,spNS /2 } and SP2={spNS /2+1,spNS /2+2, . . . ,spNS }
2. Generate all the product combinations SP′⊆SP1 with ActPay(SP′)≤WTP, sort them in an increasing order of OriPri(SP′), and store OriPri(SP′) as the list A={a1, a2, . . . , aN1 }
3. Compute a ∗∈A which is with the maximum discount rate
4. Generate all the product combinations SP′⊆SP2 with ActPay(SP′)≤WTP, sort them in a decending order of OriPri(SP′), and store OriPri(SP′) as the list B={b1, b2, . . . , bN2 }
5. Compute b ∗∈B which is with the maximum discount rate
6. argmaxOriPri(SP′)∈{a∗,b∗}DisRate(SP′)
7. Set the maximum discount number MaxDisNum= ⌊ WTP α−β ⌋ due to Lemma 3.1 8: for k=1 to MaxDisNum do
8. Initialize i=1, flag=0 and y ∗ k=(k+1)×α
9. for ai∈A do
10. j=flag+1
11. for bj∈B do
12. if ai+bj is equal to k×α then
13. y ∗ k=k×α and Break
14. else
15. if ai+bj>k×α then
16. j=j+1
17. y ∗ k= min{y ∗ k, ai+bj}
18. else
19. i=i+1
20. flag=j

21. Add SP″=argmaxOriPri(SP′)=y ∗ j DisRate(SP′) for 1≤ j≤MaxDisNum to SP∗ and refresh SP∗ by removing the combinations whose discount rates are less than that of SP″
22. Return SP

These subsets, which represent different skyline product combinations, satisfy the conditions that actual payments do not exceed the customer's willingness to pay WTP. Thereafter Lines 2 and 4 get lists A and B that store the sums of these subsets (original prices). Specially, the elements in A are sorted in increasing order while the elements in B are sorted in decending order. Here, we generate subsets and sort them all at once through merging. This can further improve the performance of TLE [24]. Lines 3 and 5 compute the elements of A and B that are equal to the original prices of the skyline product combinations with current maximum discount rate

4.2 The Lower Bound Approximate Algorithm

The LBA algorithm first removes each product *p′€SP* whose actual payment is larger than *WTP* (Line 1). Line2 initializes a list *L* with a set that contains an element "0". Thereafter, the list *L* stores original prices of candidate skyline product combinations. Lines 3-10 are applied to find candidate skyline product combinations which may bring the maximum discount rate. In Line 3, it computes *MaxDisNum* which represents the maximum discount number . Thereafter, Line 4 initializes *y€ j* ,which are the original prices of skyline product combinations that may bring the maximum discount rate without exceeding the customer's payment willingness, with ∞.

Algorithm 2 Lower Bound Approximate (LBA) Algorithm
Input: The skyline product set *SP* with |*SP*|=*NS*, a price promotion campaign "get $*β* off every $*α* purchase", a customer's payment willingness *WTP*, and a trimming parameter *ϵ* for 0<*ϵ*<1
Output: A result set *SP** of the COPC problem
Algorithm Steps:

1. Remove each product *p′∈SP* with *ActPri*(*p*)>*WTP*
2. Initialize *L*={0}
3. Set the maximum discount number *MaxDisNum*=⌊*WTPα*−*β*⌋due to Lemma 3.1
4. Initialize *y∈j*=∞ for 1≤*j*≤*MaxDisNum*
5. **while** *SP* is not empty **do**
6. *L*=*L*∪{*y*+*Ori*(*p*) : *y*∈*L*} for *p*∈*SP*
7. Sort all the elements in *L* in an increasing order and remove each element *y* from *L* if *y*−⌊*y*⌋×*β*>*WTP*
8. Compute *y*□□*L* where @*y′*∈*L*−{*y*∈*j*}, *y′*<*y*∈*j* with*y′*, *y*∈*j*□[*j*×*α*, (*j*+1)×*α*) for 1≤*j*≤*MaxDisNum* and *j*is an integer
9. Remove each element *y* from *L* which is larger than *y*□*MaxDisNum*
10. *L*=Trim(*L*−*y***j* , *ϵ*2*NS*) for 1≤*j*≤*MaxDisNum*
11. Return *SP**=*argmaxOriPri*(*SP′*)=*y***jDisRate*(*SP′*) for *j* is an integerand 1≤*j*≤*MaxDisNum*

12.Function: Trim(L, δ)
*13.*Initialize $L'=\{y1\}$
14.$last=y1$
15.for i=2 to $|L|$ **do**
16.if $yi>last\times(1+\delta)$ **then**
*17.*Append yi onto the end of L'
18.$last=yi$
*19.*Return L

Moreover, the skyline product combinations SP'' with $SP'\square SP''$ cannot be good choices for customers also. Hence, in the next iteration, it is not necessary togenerate new product combinations based on SP'. Line 10 employs a function Trim to trim some similar elements within $L-yj$. Since y contains the present optimal results, we always maintain it in L without trimming

.

4.3 The Incremental Greedy Algorithm

In this section, to further improve the performance of processing the COPC problem, we propose an incremental greedy (IG) algorithm.

Algorithm 3 Incremental Greedy (IG) Algorithm

Input: The skyline set SP of a product dataset P, a price promotion campaign "getting \$$\beta$ off every \$$\alpha$ purchase",and a customer's payment willingness WTP
Output: A result set $SP\square$ of the COPC problem
Algorithm Steps:

1. Remove each product $p\square SP$ with $ActPay(p)>WTP$
2. Initialize $PreP=1$
3. Compute product combinations $\{p\}$ where $p\square SP$ and p are with the highest discount rate, and add them to $PreP$
4. Initialize $SP=PreP$
5. Initialize $Max\ R=DisRate(\{p\})$ for $\{p\}\square PreP$
6. **while** $PreP$ is not empty **do**
7. $TempMax\ R$=0 and a set $CandSet$
8. **for** each candidate product combination $SP'\square PreP$ **do**
9. $PreP=PreP-SP'$
10. **for** each product $p\square SP-SP'$ **do**
11. Generate a new product combination $SP''=SP'\cup\{p\}$
12. **if** $ActPay(SP'')\leq WTP$ **then**
13. **if** $DisRate(SP'')>TempMax\ R$ **then**
14. $TempMax\ R=DisRate(SP'')$
15. Remove the product combinations within$CandSet$
16. Add SP'' to $CandSet$
17. **else**

18.if *DisRate*(*SP''*)=*TempMax R* **then**
*19.*Add *SP''* to *CandSet*
20.if *TempMax R*>*Max R* **then**
21.SP=*CandSet*
22.Max R=*TempMax R*
23.else
24.if *TempMax R*=*Max R* **then**
*25.SP**=*SP**∪*CandSet*
26.PreP=*CandSet*
*27.*Return *SP*`

The IG algorithm first removes all the skyline products whose actual payments are more than *WTP*. Due to the property of the COPC problem,it does not always bring a greater benefit (larger discount rate) by selecting much more products.

Complexity: The IG algorithm is composed of two stages. In the first stage (Line 1), it checks and removes the products whose actual payments are larger than *WTP*. Besides, the discount rates of each product $p<SP$ are computed.The cost of this stage is $O(NS)$. In the second stage (Lines 5 to 25), it is a while loop which generates skyline product combinations incrementally. During each iteration, it needs to combine each product combination $SP<PreP$ with a product in $SP-SP'$, the complexity of this stage is

$$O(|PreP|\times|SP-SP'|)=O(|PreP|\times|SP|).$$

V. DISCUSSIONS

In this section, we discuss variants of the COPC problem with considering other price promotion campaign and different customer's demands.

5.1 OTHER PRICE PROMOTION CAMPAIGNS

In this paper, we pay attention on the price promotion campaigns of the dependent-product selection. In the proposed algorithms, we consider one typical campaign as getting $_ off every $_ purchase. It is worth to notice that our approaches can also be used to handle the COPC problem under other campaigns of the dependent-product selection.

5.2 DIFFERENT CUSTOMER'S DEMANDS

In this paper, the COPC problem is to find the skyline product combinations *SP'*, such that they bring the maximum discount rate without exceeding the customer's payment willingness. However, when selecting products under price promotion, apart from the maximum discount rate, customers are common to have other two popular demands

that are spending or saving the most money. When customers want to spend the most money, it only needs to redefine the COPC problem in this paper by modifying the objection function as "*maximize ActPay*(*SP'*)" in Definition 5. In addition, for saving the most money (maximize the discount), the new objection function is "*maximize Discount*(*SP'*)".

5.3 SUMMARY

As analyzed above, the naive exact and TLE algorithms are appropriate to process small skyline product sets. The LBA and IG algorithms have advantages in dealing with large skyline product sets. The IG algorithm always requires far less PT with comparing to LBA. Compared to the exact algorithms and the LBA algorithm, the IG algorithm has the best scalability. For the LBA algorithm, it results in the degradation of PT with the increase of $''$ in most cases. It always needs more PT to process the Ant datasets than the Ind datasets. This is reasonable because *NS* of the Ant datasets is larger than that of the Ind datasets with equal cardinality. As *N*, *WTP*, *UDisRate* or *d* grows, it faces much more candidate skyline product combinations, and the proposed algorithms need much more PT.

VI.CONCLUSION:

In this paper, we formulate the COPC problem to retrieve optimal skyline product combinations that satisfy the customer's payment constraint and bring the maximum discount rate. To tackle the COPC problem, we propose an exact algorithm, design an approximate algorithm with an approximate bound, and develop an incremental greedy algorithm to boost the performance. We conduct a customer study to verify the significant of our COPC problem. Additionally, the experimental results on both real and synthetic datasets illustrate the effectiveness and efficiency of the proposed algorithms.

This work opens to some promising directions for future work. First, in addition to combinations of homogeneous products, we will focus on the COPC problem over products of different categories. After that, in reality, the customer's demands are diversification and individuation, and it is significant and interesting to compute optimal product combinations that meet different customer demands such as save or spend the most money under their budgets. Last but not least, we could also research top k COPC problem that aims to compute k optimal product combinations due to customer demands based on the work.

VII.REFERENCES

[1] S. B¨orzs¨onyi, D. Kossmann, and K. Stocker, "The skyline operator," in *Proc. Int'l Conf. Data Eng. (ICDE)*, pp. 421–430, 2001.[20] X. Zhou, K. Li, G. Xiao, Y. Zhou, and K. Li, "Top k favourite probabilistic products queries," *IEEE Trans. on Knowl. Data Eng*,pp. 2808–2821, 2016.

[2] Q. Wan, R. C.-W. Wong, I. F. Ilyas, M. T. O¨ zsu, and Y. Peng,"Creating competitive products," *Proc. of the VLDB Endowment*,vol. 2, no. 1, pp. 898–909, 2009.
[3] I.-F. Su, Y.-C. Chung, and C. Lee, "Top-k combinatorial skylinequeries," in *Database Systems for Advanced Applications*, pp. 79–93,Springer, 2010.
[4] Y.-C. Chung, I.-F. Su, and C. Lee, "Efficient computation of combinatorial skyline queries," *Information Systems*, vol. 38, no. 3,pp. 369–387, 2013.
[5] H. Im and S. Park, "Group skyline computation," *Information Sciences*, vol. 188, pp. 151–169, 2012.
[6] M. Magnani and I. Assent, "From stars to galaxies: skyline queries on aggregate data," in *Proc.16th Int'l Conf. on Extending Database Technology*, pp. 477–488, ACM, 2013.
[7] N. Zhang, C. Li, N. Hassan, S. Rajasekaran, and G. Das, "On skyline groups," *IEEE Trans. on Knowl. Data Eng*, vol. 26, no. 4, pp. 942–956, 2014.

