Data Science and Engineering (2019) 4:367–384
https://doi.org/10.1007/s41019-019-00106-z

Identification of *k*-Most Promising Features to Set Blue Ocean Strategy in Decision Making

Ritesh Kumar[1] · **Partha Sarathi Bishnu**[2]

Received: 18 December 2018 / Revised: 24 September 2019 / Accepted: 9 October 2019 / Published online: 22 October 2019

Abstract
Customers demand typical type of products with multiple features. We want to develop a business intelligence system which helps the company to set the blue ocean strategy by discovering *k*-most promising features (*k*-MPF) from the customers' query and a set of existing products of the similar type. In this paper, we have formulated *k*-MPF to set the blue ocean strategy with compatible features. We have experimented with our proposed algorithms using different synthetic and real datasets, and the results showed the effectiveness of our proposed algorithms.

Keywords Business intelligence · Market entry · Data mining · Decision making

1 Introduction

To retain in the market, the decision makers may use business intelligence systems to improve the manufacturer's strategic position among competitors, customers, and suppliers [1–3]. The manufacturers can collect customer choices for products and product features from their sales, e-commerce, and social networking websites [4]. Now to set the blue ocean strategy [5], the manufacturer can make intelligent use of these customer choices/query data, to decide on the features that should be selected to produce new competitive products, so that demand of the product can be maximized. To make the competition immaterial, the blue ocean strategy identifies the market space that is vast, deep, and not yet explored [5]. Using the blue ocean strategy, we can (a) build uncontested market space, (b) build and restrain new demand, (c) make the competition immaterial, and (d) break the value-cost trade-off [5]. We may construct uncontested market space by developing new products with popular features. The newly developed competitive products may attract as many customers as possible to increase the profit [4].

✉ Partha Sarathi Bishnu
psbishnu@gmail.com

Ritesh Kumar
bhritesh@gmail.com

[1] Cambridge Institute of Technology, Ranchi, India

[2] Birla Institute of Technology, Ranchi, India

We may use the products–customers-features relationship (PCFR) table (Table 1) to display the relationship between products-features and customers-features. The ep_1 to ep_5 are the existing products (EP), cq_1 to cq_{10} are the customer queries (CQ) and f_1 to f_6 are the product (sub) features (F). The content {1, 1, 1, 1, 1, 0} of the existing product ep_1 indicates that the product consists of the features $\{f_1, f_2, f_3, f_4,$ and $f_5\}$, but the feature f_6 is not present in the product ep_1. Similarly, the content {0, 1, 1, 0, 1, 1} of the customer query cq_1 indicates that the customer $c_1(\in C)$ is looking for products with the features $\{f_2, f_3, f_5$ and $f_6\}$, but the features $\{f_1$ and $f_4\}$ are not in his/her priority list. Note that a customer can put any number of queries and total number of customer queries must be more than and equal to total number of customers, but in this paper, we have used the term number of customers and the number of customers queries interchangeably.

Suppose k is set to 3, i.e., select 3 $(= k)$ most prospective features, say $\{f_i, f_j$ and $f_k\}$, where $f_i, f_j, f_k \in F$ and $i \neq j \neq k$. The promising quotient (Pf) of a sub-feature f while considering the existing products is $Pf_1 = 5/(4 + 10) = 0.3571$, where 5 is the total number of customers who are looking for the feature f_1, 4 is the total number of existing products which contain the feature f_1 and 10 is the total number of customers who inquire about the products. Similarly, the Pf_2 to Pf_6 of the features f_2 to f_6 are $10/(5 + 10) = 0.6667, 9/(1 + 10) = 0.8181, 5/(5 + 10) = 0.3333, 3/(5 + 10) = 0.2, 10/(0 + 10) = 1$, respectively, (from Table 1).

Table 1 The products–customers-features relationship table

	f_1	f_2	f_3	f_4	f_5	f_6
ep_1	1	1	1	1	1	0
ep_2	1	1	0	1	1	0
ep_3	1	1	0	1	1	0
ep_4	0	1	0	1	1	0
ep_5	1	1	0	1	1	0
cq_1	0	1	1	0	1	1
cq_2	0	1	1	0	1	1
cq_3	1	1	1	0	1	1
cq_4	1	1	1	1	0	1
cq_5	0	1	1	1	0	1
cq_6	1	1	0	0	0	1
cq_7	1	1	1	1	0	1
cq_8	0	1	1	0	0	1
cq_9	0	1	1	1	0	1
cq_{10}	1	1	1	1	0	1

Now to set the blue ocean strategy, from the calculated *Pf* values, we must select the three features (as $k = 3$) $\{f_2, f_3, \text{and } f_6\}$ to develop a new viable product. In other words, these are the three features $\{f_2, f_3, \text{and } f_6\}$ which are three most demanding features among the customers. The popularity among the customers for the feature set *kF* is *EkF* (i.e., for the features $\{f_2, f_3, \text{and } f_6\}$) $= (0.6667 + 0.8181 + 1.0 =)2.4848$. The value 2.4848 is the highest value among all the combinations consisting of 3 (k) features. The k-most (here, k is set to 3) promising features are $\{f_2, f_3, \text{and } f_6\}$.

Now, it could be possible that we cannot put these three features $\{f_2, f_3, \text{and } f_6\}$ in a single product, i.e., the features may be non-compatible (or contradict) with each other. It could be possible that the features $\{f_2 \text{ and } f_3\}$, $\{f_2 \text{ and } f_6\}$, or $\{f_3 \text{ and } f_6\}$, or all $\{f_2, f_3 \text{ and } f_6\}$ are non-compatible with each other. As an illustration of non-compatibility, assume feature f_3 (color: white) and feature f_6 (color: black) are non-compatible with each other, and in such scenario we cannot put these two features f_3 and f_6 together, because, for a single colored product, the color should be any one of the colors, either black or white, but not both (if we do not allow mixed color). In such non-compatible situation, we have to explore a *kF* set, which consists of features, which are compatible with each other so that the *EkF* is maximized. Now, such k-most promising features (k-MPF), which are identified by the business intelligence system, help the producer to set the blue ocean strategy by developing products using *kF* set features.

We have sets of existing products (*EP*) and customer queries (*CQ*) on typical type of products with product features (*F*). The promising quotient *Pf* for features f is influenced by the number of customer queries for the feature f, the total number of customer queries $|CQ|$, i.e., nc and the total number of other existing products which consist of the feature f. If the compatibility issue is there then, no simple strategy can be applied to find the best *kF* set with the highest *EkF* value. Our main objective is to develop a business intelligence system to set the blue ocean strategy by selecting k-most promising features (k-MPF) from *F*. Such set of features *kF* may guide the producers to develop new competitive products by designing a product using all the features present in *kF*.

To compare the performance of our proposed algorithms, experiments have been conducted in which all the k-MPF algorithms and the existing technique (ConsumeAttrCumul-SOC-CB-QL) of Miah et al. [6] (k-MPFsoc) and k-MPF5 with the Bayes' theorem [7] (k-MPFb) have been executed and results are compared. The k-MPF algorithms perform fairly well in almost all cases. For large dataset, the suggested technique (ConsumeAttrCumul-SOC-CB-QL) of Miah et al. [6] worked well but did not consider the compatibility issue.

The contributions of this paper are summarized as follows:

- We formulate the problem of identification of k-MPF to set a blue ocean strategy.
- Identification of highly promising features (HF), least promising features (LF), and basic features (BF) of the products.
- Three algorithms (k-MPF2, k-MPF4, and k-MPF5) are proposed to solve the k-MPF discovering problems. In k-MPF2, we use a simple greedy method where compatibility issue is not present. To address the compatibility issue, a recursive version of k-MPF algorithm k-MPF4 is implemented using backtracking concept. An iterative version of k-MPF algorithm k-MPF5 is also implemented where the *Pf* values are in sorted order. Moreover, existing techniques (ConsumeAttrCumul-SOC-CB-QL) of Miah et al. [6] (k-MPFsoc) and k-MPF5 with the Bayes′ theorem [7] (k-MPFb) have been executed for comparison purpose.

The outline of this paper is organized as follows: related work is presented in Sect. 2; in Sect. 3, we describe the proposed k-MPF algorithms. Section 4 presents the experiments. Finally, in Sect. 5, we conclude this paper.

2 Related Work

Determining the right positioning for products, potential customer finding, and the product advantage discovery are few popular microeconomics research where data mining techniques have been used [8]. Product positioning helps the producer to increase profitability by identifying the position

of a product by analyzing the customer's mind with respect to the existing products [8].

Goulding [9] discussed various aspects of new product development. The author mentioned the need for new product development, critical factors in successful new product development, the role of creativity in the new product development process, and test marketing in new product development [9].

Wan et al. [10] suggested a technique to create competitive products. Given a set of products in the existing market, they recommended a technique to create a set of "best" possible competitive products such that the newly created products are not dominated by the products in the existing market. In this paper, they did not consider customer query/choice/references for the product development process.

Wong et al. [11] proposed an algorithm to compute the skyline results efficiently using online responses for nominal data. Zhang et al. [12] proposed domination games for a modeling competition among manufacturers of a product for maximizing their expected market share. They showed that the Nash equilibrium always exists and it can be computed in polynomial time in the number of customers and products.

Michalek et al. [13] presented and demonstrated a methodology for defining a formal link between marketing product planning and engineering design decision making. Using analytical target cascading, a hierarchical optimization methodology was proposed to frame a formal optimization model which links marketing and engineering design decision-making models by defining and coordinating interactions between the two.

Miah et al. [6] introduced the problem of selecting the best attributes of a new tuple, such that this tuple would be ranked higher, given a dataset, a query log, or both, i.e., the tuple "stands out in the crowd". They suggested two algorithms (a) an optimal algorithm for small dataset and (b) three greedy algorithms for large dataset. But, in all the above mentioned techniques the compatibility issues were not considered.

Blijlevens et al. [14] discussed the knowledge of how consumers perceive product appearance by identifying appearance attributes that consumers use to distinguish the appearances of durable products.

Wu et al. [15] suggested a technique to find the qualities of the particular product. The company can promote the product by using the found qualities.

Moreover, the articles from [4, 8, 16, 17] help producer to identify the profitable products for marketing purpose. The articles from [18–20] suggested potential customers finding for marketing and advertising purpose. In this section, we have reviewed the related works, but to identify the best features no research has been proposed in consideration of the customer query and features compatibility issue.

3 Problem Statement of *k*-MPF Algorithm

In this section, we describe and define the proposed business intelligence system which helps the company to set the blue ocean strategy by selecting *k*-most promising features (*k*-MPF) from the customer queries so that a new competitive product can be developed. First, we describe the symbols used in the paper as follows:

EP	Set of existing products, $ep \in EP$
nep	Number of existing products
CQ	Set of customer queries, $cq \in CQ$
F	Set of sub-features, $f \in F$
f	The sub-feature
d	Total number of sub-features, $d = \|F\|$
k	Number of most promising features
kF	Set of k sub-features
EkF	sum of Pf values of k sub-features $f(\in kF)$
C	Set of customers, $c \in C$
NC	Total number of customers
nc	Total number of customer queries
ϕ	The feature
$\boldsymbol{\Phi}$	Set of features, $\phi \in \boldsymbol{\Phi}$
$d_{\boldsymbol{\Phi}}$	Number of features
d_ϕ	Number of sub-features of the feature ϕ and $d_\phi = \|DOM(\phi)\|$
$DOM(\phi)$	Set of sub-features of the feature ϕ
NEP_f	The total number of existing products which contain the sub-feature f
NCQ_f	The total number of customer queries about the products with sub-feature f
Pf	$NCQ_f/(NEP_f + nc)$
Bf	NCQ_f/nc
PEf	NEP_f/nep

3.1 Problem Definition

Assume a set of (potential) customers $C = \{c_1, c_2, \ldots, c_{NC}\}$, where $NC \geq 1$ is the number of customers, demanding some particular type of products. Every customer $c_i \in C$ put query $cq_i \in CQ$, about the products, where $CQ = \{cq_1, cq_2, \ldots, cq_{nc}\}$, where nc is the number of customer queries. The $\boldsymbol{\Phi} = \{\phi_1, \phi_2, \ldots, \phi_{d_{\boldsymbol{\Phi}}}\}$ is the set of features of the products, where $d_{\boldsymbol{\Phi}}$ is the number of features. According to the domain of every feature, $\phi_i \in \boldsymbol{\Phi}$ are broken into sub-features f_j, $1 \leq j \leq d_{\phi i}$, $d_{\phi i}$ is the number of sub-features of the feature ϕ_i. Total number of sub-features is $d = \sum_{i=1}^{d_{\boldsymbol{\Phi}}} d_{\phi i}$. The $EP = \{ep_1, ep_2, \ldots, ep_{nep}\}$ is the set of existing products, which are already present in the market. Here, nep is the number of existing products. From the existing products (EP) and customer queries (CQ) about certain types of products, we can construct the

products–customers-features relationship (PCFR) table (e.g., Table 1). The PCFR table contains nRow (= $|EP \cup CQ|$, i.e., $nep + nc$) rows and nCol (= $|F| = d$, F is the set of sub-features) columns. For each $ep_i, 1 \le i \le nep$, the $PCFR[i][j] = 1$, $1 \le i \le nep$, $1 \le j \le d$ indicates that the product ep_i, contains the sub-feature f_j and the $PCFR[i][j] = 0, 1 \le i \le nep$, $1 \le j \le d$, indicates that the sub-feature f_j is absent from the product ep_i.

Similarly, for each query cq_i, $1 \le i \le nc$ in CQ the $PCFR[i][j]$, $nep + 1 \le i \le nep + nc$, $1 \le j \le d$ contains the value 1 or 0. The $PCFR[i][j] = 1$ indicates that the customer c_i is looking for a product with sub-feature f_j and $PCFR[i][j] = 0$, indicates that the customer c_i is not interested in the product with the sub-feature f_j.

It is assumed that each customer $c_i \in C$ will certainly search a product with at least one sub-feature. If the customer is looking for more than one sub-features $f_j, 1 \le j \le d$, then the preferences among the sub-features are equal.

Promising Feature

A (sub) feature is promising feature, if high number of customers is looking for that (sub) feature, and at the same time availability of the products consisting of the same (sub) feature is less. To manufacture a product, if we select promising features, then the demand of the product can be met.

Definition 1 The promising quotient (Pf) of a sub-feature f is defined to be: $Pf = \frac{NCQ_f}{NEP_f + nc}$, where, NEP_f is the total number of existing products which contain the sub-feature f, and NCQ_f is the total number of customer queries about the products with sub-feature f and nc is the total number of queries. It is expected that if a sub-feature f is highly promising, its promising quotient is also high.

Now, we discuss various characteristics of the Pf. The characteristics of the promising quotient are as follows:

Characteristic (i): The value of Pf is in between 0 and 1, i.e., $0 \le Pf \le 1$. By definition $nc \ge NCQ_f$. Hence, $Pf = \frac{NCQ_f}{NEP_f + nc} \le 1$. Moreover, NCQ_f, NEP_f, nc all are non-negative quantities and the minimum value of NCQ_f be 0. Hence, the minimum value of Pf is also 0, when $NCQ_f = 0$. Hence, $0 \le Pf \le 1$.

Characteristic (ii): If $Pf = 1$, then $\frac{NCQ_f}{NEP_f + nc} = 1$. Now, $\frac{NCQ_f}{NEP_f + nc} = 1 \Rightarrow NCQ_f = NEP_f + nc$. Since, $NCQ_f \le nc$ and all these quantities are positive (≥ 0), NEP_f has to be 0 and $NCQ_f = nc$. Hence, when $Pf = 1$, $NEP_f = 0$, and $NCQ_f = nc$. This shows that if a sub-feature is demanded by all customers and the number of products satisfying this sub-feature f is not at all available in the market, the sub-feature f is the *highly promising feature* to be selected by the manufacturer.

Characteristic (iii): The scenario when a sub-feature f has least promising quotient (*least promising feature*), that is, $Pf = 0$, can be described as follows: $Pf = 0 \Rightarrow \frac{NCQ_f}{NEP_f + nc} = 0 \Rightarrow NCQ_f = 0$, that is no customer is looking for the sub-feature f. Hence, there is no question of including f in the forthcoming product.

Characteristic (iv): If two sub-features f_i and f_j are equally popular (i.e., $NCQ_{f_i} = NCQ_{f_j}$) then $Pf_i \ge Pf_j$ iff the availabilities of f_i are less than that of f_j (i.e., $NEP_{f_i} < NEP_{f_j}$). $Pf_i \ge Pf_j \Leftrightarrow \frac{NCQ_{f_i}}{NEP_{f_i} + nc} > \frac{NCQ_{f_j}}{NEP_{f_j} + nc} \Leftrightarrow NEP_{f_i} < NEP_{f_j}$ (as $NCQ_{f_i} = NCQ_{f_j}$).

Characteristic (v): If the availability of two sub-features f_i and f_j is same ($NEP_{f_i} = NEP_{f_j}$), then to a manufacturer f_i is more promising than f_j ($Pf_i > Pf_j$), if f_i is more popular among the customers than f_j. $Pf_i > Pf_j \Rightarrow \frac{NCQ_{f_i}}{NEP_{f_i} + nc} > \frac{NCQ_{f_j}}{NEP_{f_j} + nc}$ $\Rightarrow NCQ_{f_i} > NCQ_{f_j}$ (as $NEP_{f_i} = NEP_{f_j}$).

Characteristic (vi): Given that $NCQ_{f_i} > NCQ_{f_j}$, that is, ith sub-feature f_i is more popular than the jth sub-feature f_j and their popularity ratio $\frac{NCQ_{f_i}}{NCQ_{f_j}}$ be $\alpha (> 1)$, the promising factor pf_i of the sub-feature f_i is more than that of f_j, Pf_j, iff their availability ratio $\frac{NEP_{f_i}}{NEP_{f_j}}$ is less than $\alpha + \frac{nc(\alpha - 1)}{NEPf_j}$.

$Pf_i > Pf_j \Leftrightarrow \frac{NCQ_{f_i}}{NEP_{f_i} + nc} \ge \frac{NCQ_{f_j}}{NEP_{f_j} + nc} \Leftrightarrow \frac{NCQ_{f_i}}{NCQ_{f_j}} > \frac{NEP_{f_i} + nc}{NEP_{f_j} + nc} \Leftrightarrow \alpha > \frac{NEP_{f_i} + nc}{NEP_{f_j} + nc} \Leftrightarrow \alpha NEP_{f_j} - NEP_{f_i} > nc(1 - \alpha) \Leftrightarrow \frac{NEP_{f_i}}{NEP_{f_j}} < \alpha + \frac{nc(\alpha - 1)}{NEPf_j}$. Note: If the popularity ratio $\frac{NCQ_{f_i}}{NCQ_{f_j}}$ of two sub-features f_i and f_j be $\alpha (> 1)$, then their availability ratio $\frac{NEP_{f_i}}{NEP_{f_j}} < \alpha$ is a sufficient condition for f_i to be more promising than f_j.

On the basis of the above observations, we conclude the following theorem:

Theorem 1 *The definition of promising quotient is justified.*

Let $kF \subseteq F$ be a set of k sub-features which are compatible with each other. The popularity among the customers for the sub-feature set kF is EkF is defined as $\sum_{i=1}^{k} Pf_i, \forall f_i \in kF$. The strategy to select k-most promising features is by selecting k sub-features whose EkF is the maximum.

Definition 2 (k-MPF) Given a set of existing products EP, a set of customer queries CQ, a set of sub-features F, and the k-MPF (k-most promising features) are a set of k sub-features chosen from F with the maximum of EkF, where all the features $f_i, 1 \le i \le k$ in kF are compatible with each other.

Example 1 According to Table 1, the Pf_i values are $Pf_1 = 0.3571$, $Pf_2 = 0.6667$, $Pf_3 = 0.8181$, $Pf_4 = 0.3333$,

$Pf_5 = 0.2$, $Pf_6 = 1$. Assume all the sub-features are compatible to each other, hence the 3-MPF is $kF = \{f_2, f_3, f_6\}$, where $EkF = \{0.6667 + 0.8181 + 1\} = 2.4848$ which is maximum EkF value.

Definition 3 (*Highly promising feature*) Given a set of existing products EP, a set of customer queries CQ, a set of sub-features F, and the sub-feature $f_i, 1 \le i \le d$ in F are the highly promising feature (HF) (Characteristics ii), if $Pf_i \approx 1$, i.e., currently the potential customers are looking for the products with the sub-feature f_i but no or very few such products are present in the market with the sub-feature f_i. Highly promising features help the producer to set the blue ocean strategy and campaign selection [1].

Example 2 According to Table 1, the $Pf_6 = 1$, i.e., currently the customers are looking for the products with sub-feature f_6, but no such products are present in the market with the sub-feature f_6. Here, sub-feature f_6 is the highly promising sub-feature.

Definition 4 (*Least promising feature*) Given a set of existing products EP, a set of customer queries CQ, a set of sub-features F, and the sub-feature $f_i, 1 \le i \le d$ in F are the least promising feature (LF), if $Pf_i \approx 0$ (Characteristics iii), i.e., currently no or very few customers are looking for products with the sub-feature f_i. Least promising features may help the producer to set the marketing strategy.

Definition 5 (*Compatible/non-compatible features*) Given a set of sub-features F, and the sub-features f_i and f_j are in F, are compatible (or non-compatible) with each other if using these two sub-features f_i and f_j, we can (or cannot) develop a new product. In kF, all the sub-features should be compatible with each other.

We have to construct a compatibility (CMT) table to store the information about the compatibility among the sub-features.

Example 3 Let four sub-features f_1, f_2, f_3, and f_4 be there and f_1 and f_2 are non-compatible with each other. We can form kF (let $k = 3$) set using sub-features $\{f_1, f_3, f_4\}$ or using sub-features $\{f_2, f_3, f_4\}$, but in kF set we cannot keep the sub-features f_1 and f_2 together. Moreover, in the k-MPF, all the sub-features should be compatible with each other with the maximum of EkF.

Definition 6 (*Basic feature*) For all the sub-feature f_i, $1 \le i \le d$, we have to calculate Bf_i and PEf_i where $Bf_i = NCQ_{f_i}/nc$ and $PEf_i = NEP_{f_i}/nep$. If for any sub-feature $f_i \in F$, the $Bf_i \approx 1$ and $PEf_i \approx 1$ then the f_i is the basic sub-feature (BF). Basic features help the producer to develop products with common but popular sub-features.

Example 4 The Bf_i and PEf_i values of the sub-features f_1 to f_6, i.e., Bf_1 to Bf_6 (from Table 1) are 5/10 = 0.5, 10/10 = 1, 9/10 = 0.9, 5/10 = 0.5, 3/10 = 0.3, and 10/10 = 1 and PEf_1 to PEf_6 values are 4/5 = 0.80, 5/5 = 1, 1/5 =0.20, 5/5 = 1, 5/5 = 1, 0/5 = 0, respectively. Here, the PEf_i values of the sub-features f_2, f_4, f_5 are 1 and corresponding Bf_i values are 1, 0.5, and 0.3, respectively. Therefore, as per Definition 6, f_2 is the basic feature.

3.2 Construction of the Products–Customers-Features Relationship (PCFR) Table and Compatibility (CMT) Table

In this section, we discuss the construction of the products–customers-features relationship (PCFR) table and the compatibility (CMT) table as follows:

The PCFR table The *PCFR* table is a two-dimensional array with *nRow* and *nCol* number of rows and columns, respectively. The size of *nRow* is $nep + nc$ (i.e.,$|EP \cup CQ|$), and the size of *nCol* is d, i.e., the total number of sub-features. For each (nominal and ordinal) feature $\phi_i, 1 \le i \le d_{\Phi}$ (d_{Φ} is the number of features), $|DOM(\phi_i)|$ number of sub-features are there, where $DOM(\phi_i)$ is a set of sub-features of feature ϕ_i.

Note that all the sub-features $f_j, 1 \le j \le |DOM(\phi_i)|$ of feature ϕ_i are non-compatible with each other, i.e., f_k and f_l, $1 \le k, l \le |DOM(\phi_i)|, k \ne l$ are non-compatible with each other. For categorical, ordinal or Boolean data, the size of the domain of the feature is natural (i.e., as per domain of the feature or as per distinct values present in the feature) and for quantitative data, the size of the domain is user defined. The size of the nCol is $\sum_{j=1}^{d_{\Phi}} |DOM(\phi_j)|$. As we know that the features are of different types, e.g., nominal (e.g., color: red, green), ordinal (e.g., speed: high, moderate, low), Boolean (e.g., gender: male, female) or quantitative (e.g., height: 161 cm, 170 cm). For nominal, ordinal and Boolean features, the generated sub-features are based on the domain of the features or distinct values present in the features. For example, let the feature be color and for color feature the sub-features set be {white, black, silver, golden}, i.e., if the feature is color Φ_{color} and the number of distinct colors of the color feature is say black (f_b), white (f_w), silver (f_s), and golden (f_g) then $DOM(\Phi_{\text{color}})$ (= $\{f_b, f_w, f_s, f_g\}$) occupies four columns of PCFR table for four sub-features $\{f_b, f_w, f_s, f_g\}$ (Table 2) and $|DOM(\Phi_{\text{color}})| = 4$. If the feature is Boolean, e.g., mobile phone carries more than single SIM? (Φ_{sim}), then allocates two columns for yes (f_{yes}) and no (f_{no}) for the sub-features {yes and no}, respectively. If the feature is quantitative (e.g., price), then break the feature into sub-features as per user's choice. For example, the quantitative feature price ($|\Phi_p|$) may be subdivided into sub-features as follows: (1K–5K] (f_{p_1}), (5K–10K] (f_{p_2}), (10K–15K]

Table 2 Construction of the products–customers-features relationship table

	Color				Smart phone?		Single sim?		Price		
	White	Black	Silver	Golden	Yes	No	Yes	No	(1K–5K]	(5K–10K]	(10K–15K]
	$f_1(f_w)$	$f_2(f_b)$	$f_3(f_s)$	$f_4(f_g)$	$f_5(f_{\text{yes}})$	$f_6(f_{\text{no}})$	$f_7(f_{\text{yes}})$	$f_8(f_{\text{no}})$	$f_9(f_{p1})$	$f_{10}(f_{p2})$	$f_{11}(f_{p3})$
ep_1	1	0	0	0	1	0	0	1	0	1	0
ep_2	0	1	0	0	1	0	1	0	1	0	0
ep_3	0	1	0	0	1	0	0	1	0	0	1
cq_1	1	1	0	1	0	1	0	1	1	0	0
cq_2	0	1	1	1	0	1	0	1	0	0	1
cq_3	0	1	1	0	0	1	0	1	1	1	0
cq_4	0	1	0	0	1	1	1	0	1	1	1

(f_{p_3}) (Table 2). Similarly for ordinal data, e.g., feature speed ($\boldsymbol{\Phi}_s$) the distinct values are {e.g., high, moderate, and low} then the sub-features high (f_h), moderate (f_m), and low (f_l) are allocated three columns of the PCFR table.

Example 5 Table 2 demonstrates the PCFR table for the product mobile phone where four features {color, smart phone?, single SIM?, price} and eleven sub-features are there. For the feature "phone color" (nominal), the sub-features are white (f_1), black (f_2), silver (f_3), and golden (f_4). Similarly, for the features "smart phone?" (Boolean), sub-features are yes (f_5), and no (f_6), feature "carries single SIM?" (Boolean), the sub-features are yes (f_7), no (f_8), for the feature "price" (quantitative) the sub-features are (1K–5K] (f_9), (5K–10K] (f_{10}), (10K–15K] (f_{11}). In Table 2, the existing product ep_1 consists of {1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0}, i.e., the (sub)-features of ep_1 are as follows: color: white, smart phone: yes, number of SIMs more than 1: yes and the price range: more than 5K and less than and equal to 10K. The values of the customer query cq_1 are {1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0}, i.e., the customer c_1 is looking for a mobile phone with the color white or black or golden, smart phone, with more than one SIMs, and the cost of the mobile phone should be in between 1k to 5k. Note that the customers may select any number of sub-features, for example, at the same time he/she may ask for any color, say, black, or white, or silver, hence he/she can put 1 on any sub-feature (without thinking about compatibility issue). But for the existing products, the compatibility issue must be there. *Compatibility issue*: Here, sub-features f_5 and f_6 are non-compatible with each other, because the mobile phone may be smart or non-smart (non-featured) phone but not both at the same time. But (as an instance) sub-features f_1 and f_5 are compatible with each other since the white colored smart phone is possible.

Non-listed query using dynamic PCFR table: what if a customer is looking for a feature (or sub-feature) which is not enlisted in the PCFR table column. In such a scenario, we can construct a PCFR table using dynamic memory allocation where we can incorporate customer's new demand dynamically. In this research, we restrict ourselves to only static PCFR table.

The CMT table To implement compatibility and non-compatibility issue, we may construct a compatible table (CMT table). The CMT table is a symmetric matrix of order d (Table 3). If sub-feature f_i is compatible with sub-feature f_j, then the entry value of CMT [i][j] and CMT[j][i] should be zero. Similarly, if sub-feature f_i is non-compatible with sub-feature f_k, then the entry value of CMT [i][k] and CMT [k][i] should be one. Note that CMT [i][i] = 0, for all i, $1 \le i, j, k \le d$.

Example 6 If sub-feature f_1 and f_2 are non-compatible with each other, then the table entry CMT [1] [2] and CMT [2] [1] should be 1. Similarly, if sub-features f_2 and f_d are non-compatible with each other, then the table entry CMT [2][d] and CMT [d][2] should be 1 (Table 3).

3.3 Proposed *k*-MPF Algorithms

Now, we discuss different k-MPF algorithms to identify the k-MPF.

3.3.1 The *k*-MPF Algorithms Without Compatibility Issue

Algorithms 1 and 2 demonstrate the simple k-MPF algorithms. First of all, we demonstrate k-MPF1 (exhaustive search) algorithm as follows:

Table 3 Construction of CMT table

	f_1	f_2	⋯	f_d
f_1	0	1	⋯	0
f_2	1	0	⋯	1
:	:	:	⋯	:
f_d	0	1	⋯	0

Table 4 The PCFR table

	f_1	f_2	f_3	f_4	f_5
ep_1	0	1	0	0	0
ep_2	1	1	1	0	0
cq_1	0	0	1	0	1
cq_2	1	0	1	1	1

Algorithm 1: The *k*-MPF1 algorithm
Input: PCFR Table, *k*;
Output: HF, LF, BF, EkF and kF;
1. for all sub-features f_i, $1 \le i \le d$ from *F*, calculate Pf_i, Bf_i and PEf_i;
//Identify the highly promising, the least promising and the basic features.
2. if ($Pfi \approx 1$), then f_i is the highly promising feature (HF);
3. if ($Pfi \approx 0$), then f_i is the least promising feature (LF);
4. if (($PEfi \approx 1$)&&($Bfi \approx 1$)) then f_i is the basic feature (BF);
5. for ${}^dc_k$ different sub-feature sets (*kF*) of size *k*, calculate *EkF* values;
6. select *kF* set with the highest *EkF* value;
7. return HF, LF, BF, *EkF* and *kF*;

Explanation of Algorithm 1: Line 1, for all the sub-features, we calculate *Pf*, *Bf* and *PEf* values. Line 2 to 4, we identify the highly promising (HP), the least promising (LF), and the basic features (BF). Line 5, calculate EkF values for different ${}^dc_k$ number of kF sets of size *k*. Line 6, select the kF set with the maximum EkF value.

Example 7 (*k*-MPF1): The PCFR table (Table 4) is given. The *Pf* values are $Pf_1 = 1/(1+2) = 0.33, Pf_2 = 0/(2+2) = 0, Pf_3 = 2/(1+2) = 0.66, Pf_4 = 1/(0+2) = 0.5$, and $Pf_5 = 2/(0+2) = 1$. The *Bf* values are $Bf_1 = 1/2 = 0.5$, $Bf_2 = 0$, $Bf_3 = 1$, $Bf_4 = 0.5$, and $Bf_5 = 1$. The *PEf* values are $PEf_1 = 1/2 = 0.5$, $PEf_2 = 1$, $PEf_3 = 0.5$, $PEf_4 = 0$ and $PEf_5 = 0$. In summary, the *Pf*, *Bf*, and *PEf* values are as follows:

f_1	f_2	f_3	f_4	f_5
$Pf_1 = 0.33$	$Pf_2 = 0$	$Pf_3 = 0.66$	$Pf_4 = 0.5$	$Pf_5 = 1$
$Bf_1 = 0.5$	$Bf_2 = 0$	$Bf_3 = 1$	$Bf_4 = 0.5$	$Bf_5 = 1$
$PEf_1 = 0.5$	$PEf_2 = 1$	$PEf_3 = 0.5$	$PEf_4 = 0$	$PEf_5 = 0$

Here, the sub-feature f_5 is the highly promising feature and the sub-feature f_2 is the least promising feature. No basic feature is identified. Here, ${}^5c_3 = 10$ (where $d = 5$ and $k = 3$) different *kF* sets (Fig. 1) are formed and associated *EkF* values are calculated as follows:

kF	f_1,f_2,f_3	f_1,f_2,f_4	f_1,f_2,f_5	f_1,f_3,f_4	f_1,f_3,f_5
EkF	0.99	0.83	1.33	1.49	1.99
kF	f_1,f_4,f_5	f_2,f_3,f_4	f_2,f_3,f_5	f_2,f_4,f_5	f_3,f_4,f_5
EkF	1.83	1.16	1.66	1.5	**2.16**

The 3-MPF set is $\{f_3, f_4, f_5\}$ (since the *EkF* is the highest (2.16) for the sub-features $\{f_3,f_4,f_5\}$). Here, we assume that all the sub-features are compatible with each other.

***k*-MPF2 algorithm**: In the exhaustive version of the *K*-MPF algorithm (*K*-MPF1), we calculate *EkF* values for ${}^dc_k$ different sets of size *k*, hence the *k*-MPF1 algorithm is very time inefficient algorithm, therefore we propose *k*-MPF2 algorithm as follows:

Algorithm 2: The *k*-MPF2 algorithm
Input: PCFR Table, *k*;
Output: HF, LF, BF, EkF, and kF;
1. for all the sub-features f_i, $1 \le i \le d$ from *F*, calculate Pf_i, Bf_i and PEf_i and identify the highly promising (HF), the least promising (LF), and the basic features (BF);// (same as Line 1 to 4, of Algorithm 1). The *Pf* values are stored in the PFI[] array;
2. sort the sub-features *f* in ascending order of *Pf*;
3. select *k* sub-features from the right (from the sorted PFI[]) to form *kF* set;
4. return HF, LF, BF, EkF, and kF;

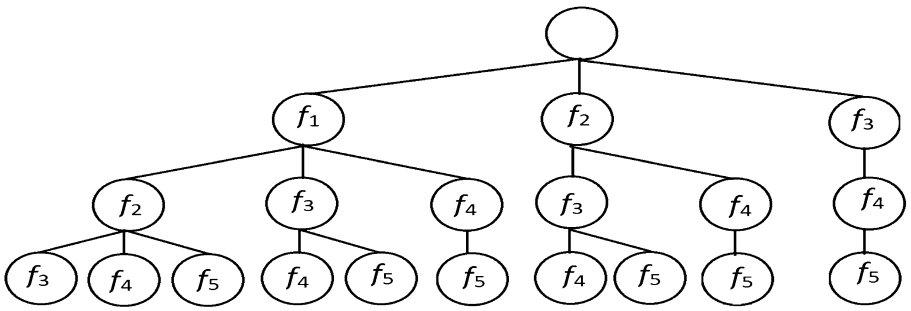

Fig. 1 The solution space of the *K*-MPF1 (Example 7)

Table 5 The PFI[] array: sub-features are arranged in ascending order of the *Pf* values

1	2	3	4	5
f_2	f_1	f_4	f_3	f_5
$Pf_2 = 0$	$Pf_1 = 0.33$	$Pf_4 = 0.5$	$Pf_3 = 0.66$	$Pf_5 = 1$

Here, in the *k*-MPF2 algorithm (Line 1), we calculate *Pf*, *Bf* and *PEf* for all the sub-features and then identify the HF, LF, and BF. In Line 2, we sort the sub-features in ascending order of calculated *Pf*, and (in Line 3), we select the best *k* numbers of the sub-features (From PFI[d] to PFI[d-k+1]).

Example 8 (*k*-MPF2) Using the PCFR table (Table 4), the *Pf*, the *Bf*, the *PEf* values are calculated and the HF, the LF, and BF are identified as Example 7. Next, the sub-features are sorted in ascending order of *Pf* values (Table 5). Select 3 (*k*) sub-features from the right side of the PFI[] (from PFI[5] to PFI[3]) (Table 5). The 3-MPF set is $\{f_3, f_4, f_5\}$ where the *EkF* value is 2.16.

3.3.2 The *k*-MPF Algorithm with Compatibility Issue

In practical situations, the compatibility issue between the sub-features is there. Therefore, we devise a new algorithm to address the compatibility issue efficiently. The motivation behind the need for new algorithms is: with compatibility issue we cannot form the *kF* set by simply selecting the sub-features according to the sorted order of the *Pf* values (as discussed in *k*-MPF2 algorithm). Therefore, to get the proper *kF* set of size *k*, we need efficient algorithm. Here, with the compatibility issue, we discuss three *k*-MPF algorithms as follows: (a) *k*-MPF3: the *k*-MPF by checking compatibility on different ${}^dc_k$ number (exhaustive) of *kF* sets, (b) *k*-MPF4: the recursive version of the *k*-MPF algorithm with backtracking technique, and (c) *k*-MPF5: the iterative version of the *k*-MPF algorithm where the *Pf* values are in sorted order.

***k*-MPF3 algorithm** In this version of *k*-MPF algorithm, we identify ${}^dc_k$ different *kF* sets of size *k* and retain few of *kF* sets where all the sub-features of the *kF* sets are compatible to one another. Out of the retained *kF* sets, the *kF* set with the maximum *EkF* value is selected. Next, we give the steps of the *k*-MPF3 algorithm as follows:

Algorithm 3: The *k*-MPF3 algorithm
Input: PCFR Table, CMT Table, *k*;
Output: HF, LF, BF, EkF, and kF;
1. for all the sub-features $f_i, 1 \le i \le d$ from *F*, calculate Pf_i, Bf_i and PEf_i and identify the highly promising (HF), the least promising (LF), and the basic features (BF); // same as Step 1 to Step 4 of Algorithm 1.

Table 6 The CMT table

	f_1	f_2	f_3	f_4	f_5
f_1	0	1	0	0	0
f_2	1	0	0	0	1
f_3	0	0	0	0	0
f_4	0	0	0	0	1
f_5	0	1	0	1	0

2. construct ${}^dc_k$ different kF sets of size *k*
3. retain the kF sets where (in the kF) the sub-features are compatible with each other;
4. select the kF set with the highest EkF value from the retained kF sets;
5. return HF, LF, BF, EkF, and kF;

Explanation of the k-MPF3 algorithm: Line 1 (Algorithm 3) is the same as Line 1 to Line 4 of Algorithm 1. Line 2, ${}^dc_k$ different *kF* sets of size *k* are generated (where *d* is the number of sub-features and *k* is the number of desired sub-features). Discard the $kF_i, 1 \le i \le^d c_k$, if f_a and f_b (where f_a, $f_b \in kF_i, a \ne b$) are not compatible with each other. Calculate *EkF* values for the retained *kF* sets. Line 4, select the *kF* set with the highest *EkF* value.

Example 9 (*k*-MPF3) The PCFR table (Table 4), the CMT table (Table 6), and $k = 3$ are given. The *Pf*, *Bf*, and *PEf* values are calculated as Example 7. Here, ${}^5c_3 = 10$ (where $d = 5$ and $k = 3$) different *kF* sets are formed. If the sub-features of *kF* are compatible with each other, then the *kF* is retained and associated *EkF* values are calculated as follows:

kF	f_1,f_2,f_3	f_1,f_2,f_4	f_1,f_2,f_5	f_1,f_3,f_4	f_1,f_3,f_5
EkF	–	–	–	1.49	**1.99**
kF	f_1,f_4,f_5	f_2,f_3,f_4	f_2,f_3,f_5	f_2,f_4,f_5	f_3,f_4,f_5
EkF	–	1.16	–	–	–

Here, using *k*-MPF3, three *kF* $\{f_1,f_3,f_4\}$, $\{f_1,f_3,f_5\}$, $\{f_2,f_3,f_4\}$ sets are retained (where the sub-features are compatible with each other) and associated *EkF* values are calculated. The highest *EkF* value is 1.99 for the sub-features $\{f_1,f_3,f_5\}$, hence the 3-MPF is $\{f_1, f_3, f_5\}$.

The complexity of *k*-MPF3 algorithm is $O({}^dc_k \times {}^kc_2)$ (Total ${}^dc_k \times {}^kc_2$ different compatibility checking are there), therefore we need a competent algorithm to address the compatibility issue.

***k*-MPF4 Algorithm** First of all, we present the *k*-MPF4 algorithm as follows:

Algorithm 4: The k- MPF4 algorithm
Input: PCFR Table, CMT Table, k;
Output: HF, LF, BF, EkF, and kF;
1. KFTemp = {}, EKFTemp = {}, PFI ={}, Tag =0; // Global variables
2. for all sub-features f_i, $1 \leq i \leq d$ from F, calculate Pf_i, Bf_i and PEf_i and identify the highly promising (HF), the least promising (LF) and the basic features (BF);
3. $kF = \{\}$;
4. for (i = 1; i <= d - k + 1; i = i+1){ // d is the number of sub-features
5. MPFCE(kF, i, k); // Call function MPFCE()
6. }
7. if (Tag == 0)
8. print ("No solution exist with present k value, select lesser k value");
9. else return HF, LF, BF, EkF, kF;

// Function MPFCE
10. **MPFCE(kF, i, k)** {
11. select the i^{th} sub-feature say (f_x); //f_x = PFI[i]
12. if ($|kF| != 0$) {
13. for all the sub-features f_y present in kF {
14. //check the compatibility between f_x with f_y
15. if (CMT[x][y] == 1) then return; // Backtrack
16. }
17. }
18. kF = union(kF, f_x);
19. if ($|kF| == k$) {
20. Tag = 1;
21. store kF and EkF values in KFTemp and EKFTemp respectively;
22. return; //Backtrack
23. }else{
24. if (i == d) then return;
25. for (p = i+1; p <= d - (k-|kF|-1); p = p+1) {
26. MPFCE(kF, p, k); // Call MPFCE() recursively
27. }
28. }
29.}

Explanation of the k-MPF4 algorithm (Algorithm 4): Line 1, four global variables KFTemp, EKFTemp, PFI, and Tag are declared. Line 4, in for loop, i varies from 1 to $(d - k + 1)$. Line 5, algorithm calls the MPFCE() function where kF, i and k are passed as the parameters. Line 10 to 29, the function MPFCE() is defined. Line 12 to 17, compatibility in between ith, i.e., f_x with sub-feature(s) present in the kF set is (are) checked. Line 15, if f_x and f_y are the non-compatible with each other, then the formation of the kF set is stopped and return back (backtrack) to check the compatibility of the $(i + 1)$th sub-feature with the sub-features present in the kF set. Line 19 to 23, if the size of the kF set is same as k then update the KFTemp and the EKFTemp to store the kF and associated EkF value, respectively, and the Tag variable is set to 1. Line 22, return back to form another kF set with $(i + 1)$th sub-feature. Line 25, call the MPFCE() function to check the remaining $(p + 1)$th sub-features. Note that in Line 25, p varies from $(i + 1)$ to $d - (k - |kF| - 1)$, hence algorithm reduces the number of kF sets formation. Line 7, if Tag is 1, then the kF with maximum EKF is returned.

The complexity of k-MPF4 algorithm is $O({}^{d}c_k \times {}^{k}c_2)$ (maximum total ${}^{d}c_k \times {}^{k}c_2$ compatibility checking are there) in the worst case situation and $\Omega({}^{(d-k+2)}c_2)$ (i.e., ${}^{(d-k+2)}c_2$ compatibility checking are there) in the best case situation.

Example 10 (k-MPF4) The PCFR table (Table 4), the CMT table (Table 6), and $k = 3$ are given. Figure 2 demonstrates the solution space of the k-MPF4. At each node, we check the compatibility to block the paths. To guide the search at each node, we check only those successors' nodes that are compatible with its ancestor's nodes. Initially, the partial solution (kF) is empty and as each new node is visited, the partial solution is extended. Line 4, i varies from 1 to $5 - 3 + 1 = 3$. Line 5, function MPFCE() is called and the parameters $kF = \{\}$, $i = 1$, and $k = 3$ are passed. Line 11, $i = 1$, hence f_1 is selected

from the PFI[1] (since the Pf_i values are stored in PFI[] array). The leftmost subtree contains the solution with sub-feature f_1 (Fig. 2). Line 12, the *if* statement is false since the size of the *kF* is 0. Line 18, the *kF* is updated, hence partial $kF = \{f_1\}$. Line 19, the *if* statement is false since the size of the *kF* is 1. Line 25, p varies from 2 to 4, and the function MPFCE() is called (recursively), where $p = 2$, $k = 3$, and $kF = \{f_1\}$ are passed as parameters. Line 11, $i = 2$, hence f_2 is selected from the PFI[2]. Line 12, the *if* statement is true since the size of *kF* is 1. Line 13, sub-feature f_1 and sub-feature f_2 are non-compatible (Table 6: The CMT table) with each other; hence, we do not explore further, and the f_2 node is killed (and its decedents sub-features (by showing faded line)) and return to Line 25, (of previously called MPFCE() function), p is incremented to 3, and the function MPFCE() is called, where $p = 3$, $k = 3$, and $kF = \{f_1\}$ are passed as parameters. Similarly, the f_3 (sub-feature) node is explored, since f_3 is compatible with f_1, the updated *kF* set to $\{f_1, f_3\}$. Next, the compatibility between $\{f_1, f_3\}$ with sub-feature f_4 is checked successively (Line 12 to 16). Sub-feature f_4 is compatible with $\{f_1$ and $f_3\}$; hence, f_4 is included in the *kF*. Now the $kF = \{f_1, f_3, f_4\}$ and *EkF* value is calculated. After visiting three (k) nodes, the depth search is blocked and the node f_4 has no unfinished descended. The depth first search now return back to explore for other solutions, i.e., check the compatibility $\{f_1, f_3\}$ with sub-feature f_5. Since f_5 is compatible with the elements of the $kF = \{f_1, f_3\}$ set; hence, f_5 is included in the *kF* set and the corresponding *EkF* value is calculated. The same process is carried out for the rest of the nodes (Fig. 2). Finally, identify the maximum *EkF* value (1.99) and the associated $kF = \{f_1, f_3, f_5\}$ set to get the k-most promising features.

***k*-MPF5 algorithm** Now, we discuss iterative version of k-MPF (k-MPF5) algorithm with compatibility issue. In k-MPF5, number of *kF* sets formation can be reduced more efficiently. The algorithmic steps of the k-MPF5 are as follows:

Algorithm 5: The k- MPF5 algorithm
Input: PCFR Table, CMT Table, k;
Output: HF, LF, BF, EkF, kF;
1. for all the sub-features f_i, $1 \leq i \leq d$ from F, calculate Pf_i, Bf_i and PEf_i and identify the HF, LF and BF; // same as Step 1 to 4 of Algorithm 1.
2. sort the sub-features f_i in descending order of Pf where S[] stores the sorted Pf values and I[] stores original index values of Pf;
3. for (i = 1; i<=d-(k-1); i++){
4. for (j = i; j<=d; j++){
5. if (CMT[I[1,i],I[1,j]]) == 1 then AR[i,j] = -1; GR[i,j] =-1;
6. else then AR[i,j] = S[1,j]; GR[i,j] = I[1,j];
7. }}
8. EkF = 0; kF={}; Tag = 0;
9. for (i = 1; i < = d-(k-1); i++) {
10. tkF = i; tEkF =AR[i,i]; IND = i;
11. for (j = i + 1; j < = d; j++){
12. if(AR[i,j]> -1){
13. IND = union(IND,j); // insert j into IND
14. tEkF += AR[i,j];
15. tkF = union(tkF,GR[i,j]); //insert jth feature into tkF;
16. for (m = j+1; m<=d; j++){
17. if (AR[j,m] == -1) then AR[i,m] = -1;
18. }
19. if (EkF<tEkF){
20. EkF = tEkF;
21. kF = tkF;
22. }
23. if (k == |tkF|){
24. if((i*k+k*(k-1)/2) == sum(IND)) then Tag = 1;
25. break;
26. }
27. }
28. }if (Tag == 1) then break;
29. }
30. return HF, LF, BF, EkF, kF;

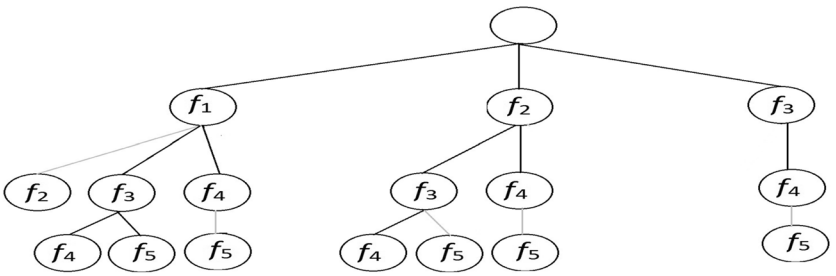

Fig. 2 The solution space of the *K*-MPF4 (Example 10)

Explanation of the k-MPF5 algorithm (Algorithm 5): In Line 1, *k*-MPF5 identifies HF, LF, and BF. Line 2, the sub-features *f* are sorted in descending order of *Pf*, where S[] stores the sorted *Pf* values and I[] stores sub-features number (or index of the sub-features) accordingly. Line 3 to 7, two (two dimensional) arrays AR[] and GR[] (of size $d-(k-1)$ by d and upper-triangular) are constructed. In AR[], we store the sorted *Pf* values (from the S[], which contains the sorted Pf values) which are compatible with the *i*th row sub-feature, and in GR[] we store the associated index of the sub-features which are stored in AR[]. In both the arrays AR[i ,j] and GR[i, j], if the *i*th sub-feature is not compatible with the *j*th sub-feature then − 1 is inserted. Line 8, the variables EkF = 0, kF = { }, and Tag = 0 are initialized. Line 9, *for-loop* executes $d-(k-1)$ times. Line 10, the *i* value is inserted into tkF and IND sets. The tEkF is initialized with AR[i,i] value. Line 11, *j* varies from $i+1$ to *d*. For every *i*th sub-feature, the compatibility (Line 12) is checked with other sub-features (*j*) and if they are compatible with each other (sub-feature *i* with sub-feature *j*) then IND, tEkF, and tkF are updated. Line 16 to 18, after every inclusion of the *j*th sub-features, the AR[] table is updated (if needed). If any element in between $(j+1)$ to *d* column of the *j*th row is − 1, then the *i*th row is updated accordingly. In other words, if AR[j, m] is − 1, then AR[i, m] is also updated to − 1. Line 19 to 22, maximum EkF value and associated kF set are captured. Line 23, if *k* number of features are identified then next *i*th values are checked. Line 24, if index of all the *k* sub-features is adjacent then the algorithm immediately returns the output, since the possibility to get the better result is nil.

The time complexity of *k*-MPF5 algorithm is $O(d^3-d^2k)$ in the worst case situation and $\Omega(d^2)$ in the best case situation, but note that in Line 2, the algorithm needs extra time ($O(dlogd)$) for sorting.

Example 11 (*k*-MPF5): The sub-features are arranged in descending order of the *Pf* values. The order is as follows: $Pf_{5(1)}$, $Pf_{3(2)}$, $Pf_{4(3)}$, $Pf_{1(4)}$, and $Pf_{2(5)}$ and the ordered Pf_i values are (S[] =){1, 0.66, 0.5, 0.33, 0}, respectively (Line 2 of Algorithm 5). Here, the meaning of $Pf_{i(j)}$ is the *i*th feature and the *j*th order. The I[] stores the original index of the sub-features, i.e., I[] = {5, 3, 4, 1, 2}. Line 3 to 7, AR (Table 7) and GR (Table 8) are formed.

Line 8, EkF = 0, kF = { }, and Tag = 0 are initialized. Line 9, *i* varies from 1 to (5 − (3 − 1) =)3. When *i* = 1, (Line 10) tkF is initialized with *i* (=1), tEkF is initialized with A[1, 1] (i.e., 1), and IND is initialized with *i*(= 1). Line 11, *j* varies from 2 to 5. When *j* = 2, (Line 12) AR[1, 2] is 0.66 which is more than − 1, hence, *j*(= 2) is included in IND , tEkF is (1 + 0.66 =)1.66, tkF is {1, 2}. Line 16, variable *m* varies from 3 to 5, no value in 2nd row is − 1 hence, first row is not updated. Line 19 and 22, EkF and kF are updated. Line 23, the *if* condition is false, since present size of the tkF is 2 not 3 (*k*). Next, *j* = 3 (Line 11), AR [1, 3] is − 1 (Line 12), the *if* condition is false. Next, *j* = 4 (Line 11), AR[1, 4] = 0.33, Line 12, the *if* condition is true. The IND = {1, 2, 4}, the tEkF = 1.99, and tkF = {5, 3, 1} are updated. Line

Table 7 The initial AR table

	f_5	f_3	f_4	f_1	f_2
f_5	1	0.66	− 1	0.33	− 1
f_3	0	0.66	0.50	0.33	0
f_4	0	0	0.50	0.33	0

Table 8 The GR table

	1	2	3	4	5
1	5	3	− 1	1	− 1
2	0	3	4	1	2
3	0	0	4	1	2

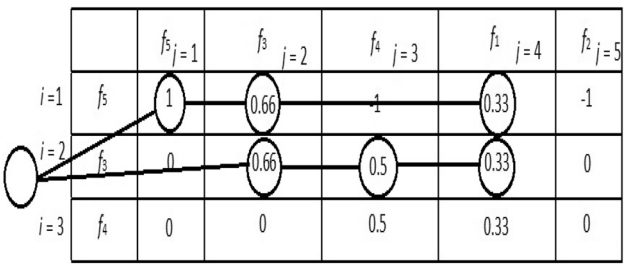

Fig. 3 The solution space of the *K*-MPF5 (Example 11)

16, m varies from 5 to 5, the AR[4, 5] is − 1 hence, AR[1, 5] is updated to − 1. Line 19 and 22, EkF = 1.99 and kF = {5, 3, 1} are updated (maximum EkF value and associated kF set are captured). Line 23, the *if* condition is true (since size of the tkF is 3(k)). Line 24, the if condition is false, since $i * k + k * (k-1)/2$, i.e., $1 * 3 + 3 * (3-1)/2 = 6$ is not equal to sum of the content of IND, i.e., $(1 + 2 + 4) = 7$ (Fig. 3).

Next $i = 2$ (Line 9), in Line 10, tkF = {3}, tEkF = 0.66, and IND = {2}. The j varies from 3 to 5 (Line 11). If $j = 3$, AR[2, 3] is 0.5, the *if* condition is true (Line 12), hence IND = {2, 3}, tEkF = {1.1666}, and tkF = {3, 4} are updated. Line 16, m varies from 4 to 5, no elements of third row is − 1, hence, second row is not updated. Line 19, the *if* condition is false (since 1.99 > 1.16). Line 23, the *if* condition is false (since |kF| = 2).

If $j = 4$, AR[2, 4] is 0.33, the *if* condition is true, hence, IND = {2, 3, 4}, tEkF = {1.49}, and tkF = {3, 4, 1} are updated. Line 16, m varies from 5 to 5, AR[4, 5] is − 1, hence, AR[2,5] is updated to − 1. Line 19, the *if* condition is false (since, 1.99 > 1.49). Line 23, the *if* condition is true (since, |kF| = 3), Line 24, the *if* condition is true, since $i * k + k * (k-1)/2$, i.e., $2 * 3 + 3 * (3-1)/2 = 9$ is equal to sum of the content of IND, i.e., $(2 + 3 + 4) = 9$. Line 30, HF, LF, BF, EkF = 1.99 and kF = {5, 3, 1} are returned (Fig. 3).

4 Experiments

The experiments were conducted on a PC with an Intel(R) Core(TM), i5-4570 processor, (3.20 GHz), and 8 GB RAM running the Windows 10 operating system. All the algorithms have been coded in Octave-3.2.4. GNU Octave (http://www.gnu.org/software/octave) is a high-level interpreted programming language for numerical computation. Using $k = d_{\Phi}$ number of sub-features, we can develop a new product. To show the effectiveness of the new product, we use the performance parameter as follows: a product must contain minimum number (say, d_{Φ}) of features to manufacture. Hence, we may select d_{Φ} sub-features (with compatibility) to generate a synthetic product. Intuitively, in present time, the generated synthetic product may fulfill as many customers (higher PNP_n value) desire as possible. For the jth existing product, we calculate number of preference matching (i.e., matching 1's) nom_{ij} with all the existing customers $i, 1 \le i \le nc$. The performance of jth existing product is $PEP_j = \frac{\sum_{i=1}^{nc} nom_{ij}}{nc}$. Next, we calculate $mPEP = \max_{1 \le j \le nep} PEP_j$. The $mPEP$ value of an existing product exhibits the maximum number of customers which are satisfied with the (existing) product. For the new product, we calculate performance of new product $PNP_n = \frac{\sum_{i=1}^{nc} nom_{inew}}{nc}$. We compare PNP_n with $mPEP$. If $PNP_n > mPEP$, then the new (synthetic) product can be the best alternative than the existing products (since, the new product may fulfill the current demand of the customers), but if $PNP_n \le mPEP$, then there is no need to develop a new product, since the best product is already present in the market.

An existing technique ConsumeAttrCumul-SOC-CB-QL of Miah et al. [6] (k-MPFsoc) has been executed to select the k sub-features from the customer queries and existing products information. For large dataset, the suggested technique (ConsumeAttrCumul-SOC-CB-QL) by the authors Miah et al. [6] works well, but they did not consider the compatibility issue. Hence, for comparison purpose, we have incorporated the compatibility issue in their ConsumeAttrCumul-SOC-CB-QL technique. Moreover, we have implemented K-MPF5 with the Bayes' theorem [7] (k-MPFb) for comparison purpose. For every sub-feature $f_i, 1 \le i \le d$, we calculate, $P(f_i|1) = \frac{P(f_i)P(1|f_i)}{\sum_{i=1}^{d} P(f_i)P(1|f_i)}$ (posterior probability), where, $P(f_i) = \frac{1}{d}$ (prior probabilities), $P(1|f_i) = \frac{NO1_i}{|nc+nep|}$ (conditional probabilities), $NO1_i$ is the number of 1 present in f_i sub-feature. It is expected that if a sub-feature f_i is highly promising, its $P(f_i|1)$ is also high. To implement the k-MPFsoc and the k-MPFb, we inverted the content of the EP and merge with CQ to construct PCFR. For comparison purpose, we have calculated EkF values on both the techniques k-MPFb and k-MPFsoc separately.

We have executed seven programs which are as follows: (a) *Without considering the compatibility issue*: (i) k-MPF1 (exhaustive), (ii) k-MPF2 (proposed 1), (b) *Considering the compatibility issue* (i) k-MPF3 (exhaustive), (ii) k-MPF4 (proposed 2) (iii) k-MPF5 (proposed 3), (c) k-MPFb (using Bayes' Theorem with k-MPF5), (d) k-MPFsoc (using [6]). For each experiment, three runs have been executed and the minimum execution time has been reported.

4.1 Datasets

To show the effectiveness of our proposed algorithms, we have used two real datasets auto data and car data from the UCI machine learning repository: (http://archive.ics.uci.edu/ml/).

Auto data The total number of data is 398. After removing the data with missing values, the number of data is 392. The number of features is 9. The features are as follows: (i) mpg (continuous), (ii) cylinders (multi-valued discrete), (iii) displacement (continuous), (iv) horsepower (continuous), (v) weight (continuous), (vi) acceleration (continuous), (vii) model year (multi-valued discrete), (viii) origin (multi-valued discrete), (ix) car name (string (unique for each instance)). For our experiment, we have selected first six features. First feature mpg, which is a continuous type,

the minimum value is 9 and the maximum value is 46.60. We have divided this feature ϕ_1 into two sub-features as follows: f_1 is the first sub-feature for the range (9–25] and f_2 is the 2nd sub-feature for the range (25–50].

The second feature is cylinders ϕ_2 of type multi-valued discrete (ordinal), the minimum value is 3 and the maximum value is 8. We have divided this feature ϕ_2 into three sub-features as follows: f_3 is the third sub-feature and the range is [3, 4], f_4 is the fourth sub-feature and the range is [5, 6], and f_5 is the fifth sub-feature and the range is [8, 9].

The third feature is displacement ϕ_3 of type continuous, the minimum value is 68 and the maximum value is 455. We have divided this feature ϕ_3 into two sub-features as follows: the f_6 is (68–200], the f_7 is (200–455]. The fourth feature is horsepower ϕ_4 of type continuous, the minimum value is 46 and the maximum value is 230. We have divided this feature into three sub-features as follows: the f_8 is (46–100], the f_9 is (100–199], and the f_{10} is (199–230]. The fifth feature is weight ϕ_5 of continuous type, the minimum weight is 1613 and maximum weight is 5140. The divisions of the feature ϕ_5 are as follows: the f_{11} is (1613–3000], the f_{12} is (3000–5140].

The sixth feature is acceleration ϕ_6 of type continuous, the minimum value is 8 and the maximum value is 24. We have divided this feature ϕ_6 into three sub-features as follows: the f_{13} is (0–10], the f_{14} is (10–20], and the f_{15} is (20–30]. From the auto data, we have selected the data iteratively and the values are compared with sub-features range and the PCFR table of auto data is filled with 0 and 1. For example, the first data (or row) of auto data are {18.0, 8, 307.0, 130.0, 3504.0, 12.0} of features {ϕ_1, ϕ_2, ϕ_3, ϕ_4, ϕ_5, ϕ_6}, respectively. Therefore, the first row of the PCFR table is {1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0} of sub-features {f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10}, f_{11}, f_{12}, f_{13}, f_{14}, f_{15}}, respectively.

Similarly, the tenth data are {15.0, 8, 390.0, 190.0, 3850.0, 8.5} and hence the tenth row of the PCFR table is {1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0}. Number of rows and columns of the PCFR table of auto data are 392 and 15 (sub-features), respectively. We have constructed a CMT table which is a square matrix of order 15. Under the first feature ϕ_1, two sub-features f_1, and f_2 are non-compatible with each other. Similarly, under the second feature ϕ_2, three sub-features f_3, f_4, and f_5 are non-compatible with each other. But f_1 with {f_3, or f_4, or f_5} and f_2 with {f_3, or f_4, or f_5} are compatible with each other.

Car data The total number of data is 1728, and the number of features is 6. The features are as follows: (i) buying (very-high, high, med, low), (ii) maintenance (very-high, high, med, low), (iii) doors (2, 3, 4, 5 or more), (iv) persons, (2, 4, more), (v) luggage-boot (small, med, big), and (vi) safety (low, med, high). All the features are ordinal types. From first feature buying ϕ_1, we have constructed four sub-features f_1, f_2, f_3, and f_4 for (buying price) very-high, high, med, low, respectively. From the second feature maintenance ϕ_2, we have constructed sub-features f_5, f_6, f_7, and f_8, for (maintenance) very-high, high, medium, and low, respectively.

In this way, we have constructed the PCFR table for the car data using all the sub-features. Number of rows and columns of the PCFR table of car data are 1728 and 21 (sub-features), respectively. From car data, we have selected the data iteratively and compared with sub-feature values and the PCFR table is filled with 0 and 1. For example, the first data is {vhigh, vhigh, 2, 2, small, low}. Therefore, the first row of the PCFR table is {0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0}.

For all the data, from PCFR table, we have selected first thirty percent data as existing products (*EP*) and remaining seventy percent data as customer queries (*CQ*).

Synthetic data We have used two synthetic data [21] (SYN1 (correlated) and SYN2 (uncorrelated)) to construct the PCFR table. The number of data are 1000, and 2000, dimensions (sub-features) are 20 and 40, features are 7 and 11, respectively. Moreover, we have generated two CMT tables which are the square matrix of order 20 and 40, respectively. Moreover, to show the scalability of our proposed algorithms, we have generated synthetic data (The PCFR table) of different sizes. We have executed all the algorithms on different (i) *dimensions* (*d*): here, number of

Table 9 The *Pf*, *Bf*, and *PEf* values of the auto data

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
Pf	0.2393	0.6044	0.5078	0.2278	0.1419	0.5758	0.2587	0.5666
Bf	0.3054	0.6945	0.5854	0.2436	0.1709	0.6763	0.3236	0.6800
PEf	0.6495	0.3504	0.3589	0.1623	0.4786	0.4102	0.5897	0.4701
	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	
Pf	0.2699	0	0.5290	0.3000	0	0.6700	0.0645	
Bf	0.3200	0	0.6291	0.3709	0	0.9345	0.0654	
PEf	0.4358	0.0941	0.4444	0.5555	0.0854	0.8803	0.0341	

Table 10 The *Pf*, *Bf*, and *PEf* values of the car data

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}
Pf	0.3570	0.3570	0.2669	0	0.2458	0.2458	0.2458	0.1695	0.2208	0.2208	0.2208
Bf	0.3570	0.3571	0.2859	0	0.2677	0.2677	0.2677	0.1966	0.2454	0.2454	0.2454
PEf	0	0	0.1660	0.8339	0.2084	0.2084	0.2084	0.3745	0.2606	0.2604	0.2606
	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}	f_{21}	
Pf	0.2411	0.2886	0.2932	0.2932	0.2904	0.2914	0.2932	0.2914	0.2914	0.2923	
Bf	0.2636	0.3305	0.3347	0.3347	0.3322	0.3331	0.3347	0.3332	0.3335	0.3338	
PEf	0.2181	0.3397	0.3301	0.3301	0.3359	0.3339	0.3301	0.3339	0.3339	0.3320	

Table 11 Results of *k*-MPF1 and *k*-MPF2 on different *k* of the auto data

No.	*k*	*EkF*	T (*k*-MPF1)	T (*k*-MPF2)	*kF* (sub-features)
1	2	1.2843	0.1934	0.1124	2, 14
2	3	1.8602	0.2969	0.1875	2, 6, 14
3	4	2.4268	0.4218	0.2281	2, 6, 8, 14
4	5	2.595	0.8125	0.2593	2, 6, 8, 11, 14
5	6	3.4638	1.0937	0.2906	2, 3, 6, 8, 11, 14
6	7	3.7638	1.5157	0.3562	2, 3, 6, 8, 11, 12, 14
7	8	4.0337	1.7188	0.4160	2, 3, 6, 7, 8, 11, 12, 14
8	9	4.2924	1.8123	0.4375	2, 3, 6, 7, 8, 9, 11, 12, 14
9	10	4.5318	1.9314	0.4531	1, 2, 3, 6, 7, 8, 9, 11, 12, 14

rows is 100 and *k* is 3 for all the cases, (ii) *sizes* (*nep* + *nc*): here, the number of sub-features (columns) is 10 (*d*) and the *k* value is 3 for all the cases, and (iii) *k*: for all the different *k* values, the number of rows is 5000 and the number of sub-features is 20. Moreover, to execute all the *k*-MPF algorithms, we have generated CMT tables randomly using Octave programming. Note that for synthetic data [21], we have directly generated (using 1's and 0's) the PCFR and the CMT tables through programming.

4.2 Results and Analysis

Now, we discuss the analysis of the different outcomes of all the algorithms as follows

Auto data: The *Pf*, *Bf*, and *PEf* values of the Auto data are given in Table 9. The *Pf* value of 10th and 13th sub-features is 0, and hence, we can say that the f_{10} and the f_{13} sub-features are the least promising features and the *Pf* value of the 14th feature is 0.67 which is the maximum and hence we can conclude that sub-feature f_{14} is the highly promising feature. We have applied *k*-MPF1, *k*-MPF2, *k*-MPF3,

Table 12 Results of *k*-MPF3, *k*-MPF4, and *k*-MPF5 on different *k* of the auto data

No.	*k*	*EkF*	T (*k*-MPF3)	T (*k*-MPF4)	T (*k*-MPF5)	*kF* (sub-features)
1	2	1.2843	0.3335	0.2543	0.2804	2,14
2	3	1.8602	0.4500	0.3125	0.3010	2, 6, 14
3	4	2.4268	0.9350	0.7501	0.4724	2, 6, 8, 14
4	5	2.9559	3.5470	1.7032	0.4930	2, 6, 8, 11, 14
5	6	3.4638	8.1450	3.2545	0.9702	2, 3, 6, 8, 11, 14

Table 13 Results of *k*-MPF1 and *k*-MPF2 on different *k* of the car data

No.	*k*	*EkF*	T (*k*-MPF1)	T (*k*-MPF2)	*kF* (sub-features)
1	2	0.7140	1.2032	0.9111	1, 2
2	3	1.0073	1.2032	1.0111	1, 2, 14
3	4	1.3006	1.8423	1.2502	1, 2, 14, 15
4	5	1.5938	3.8751	1.2800	1, 2, 14, 15, 18
5	6	1.8862	8.9594	1.2903	1, 2, 14, 15, 18, 21
6	7	2.1776	20.5187	1.3216	1, 2, 14, 15, 17, 18, 21
7	8	2.4690	34.2543	1.4821	1, 2, 14, 15, 17, 18, 19, 21
8	9	2.7604	44.5521	1.4922	1, 2, 14, 15, 17, 18, 19, 20, 21
9	10	3.0508	71.6663	1.5232	1, 2, 14, 15,16, 17, 18, 19, 20, 21

Table 14 Results of *k*-MPF3, *k*-MPF4, and *k*-MPF5 on different *k* of the car data

No.	*k*	*EkF*	T (*k*-MPF3)	T (*k*-MPF4)	T (*k*-MPF5)	*kF* (sub-features)
1	2	0.6502	0.3200	0.3200	0.2980	1, 14
2	3	0.9435	0.4000	0.3700	0.3001	1, 14, 18
3	4	1.2359	0.8910	0.7039	0.8765	1, 14, 18, 21
4	5	1.4817	3.1320	2.2300	2.9110	1, 5, 14, 18, 21
5	6	1.7228	10.440	7.4450	7.0900	1, 5, 12, 14, 18, 21

Table 15 Results of *k*-MPF1 and *k*-MPF2 on different *k* of the SYN1 data

No.	*k*	*EkF*	T (*k*-MPF1)	T (*k*-MPF2)	*kF* (sub-features)
1	2	0.8902	3.8564	1.0011	4, 16
2	3	1.3222	4.8767	1.2311	4, 8, 16
3	4	1.7538	5.8773	1.2222	4, 6, 8, 16
4	5	2.1854	6.8751	1.9876	4, 6, 8, 11, 16
5	6	2.6158	7.0023	2.1903	4, 6, 7, 8, 11, 16
6	7	3.0424	7.5187	2.3216	4, 6, 7, 8, 11, 12, 16
7	8	3.4675	9.2543	3.4821	3, 4, 6, 7, 8, 11, 12, 16
8	9	3.8917	12.5521	3.4922	3, 4, 6, 7, 8, 11, 12, 16, 18
9	10	4.3103	17.6663	3.5232	3, 4, 6, 7, 8, 11, 12, 13, 16, 18

Table 16 Results of *k*-MPF3, *k*-MPF4, and *k*-MPF5 on different *k* of the SYN1

No.	*k*	*EkF*	T (*k*-MPF3)	T (*k*-MPF4)	T (*k*-MPF5)	*kF* (sub-features)
1	2	0.8902	0.8912	0.3200	0.7302	4, 16
2	3	1.3219	2.8010	2.7700	1.8956	4, 6, 16
3	4	1.7535	3.0510	4.7039	2.2544	4, 6, 11, 16
4	5	2.1720	6.1020	5.8300	5.0010	4, 6, 11, 13, 16
5	6	2.5710	7.5930	8.445	8.1991	2, 4, 6, 11, 13, 16

Table 17 Results of *k*-MPF1 and *k*-MPF2 on different *k* of the SYN2 data

No.	*k*	*EkF*	T (*k*-MPF1)	T (*k*-MPF2)	*kF* (sub-features)
1	2	0.8813	14.0234	10.1127	17, 21
2	3	1.3081	14.2032	10.1910	17, 21, 39
3	4	1.7327	15.9103	10.3502	3, 17, 21, 39
4	5	2.1522	16.9911	10.9090	3, 17, 21, 27, 39
5	6	2.5709	17.0001	11.0001	3, 17, 21, 25, 27, 39
6	7	2.9892	17.8187	11.2212	3, 9, 17, 21, 25, 27, 39
7	8	3.4075	18.9054	11.9987	3, 8, 9, 17, 21, 25, 27, 39
8	9	3.8251	22.2227	12.0212	3, 8, 9, 11, 17, 21, 25, 27, 39
9	10	4.2402	29.0098	12.2211	3, 6, 8, 9, 11, 17, 21, 25, 27, 39

Table 18 Results of *k*-MPF3, *k*-MPF4 and *k*-MPF5 on different *k* of the SYN2

No.	*k*	*EkF*	T (*k*-MPF3)	T (*k*-MPF4)	T (*k*-MPF5)	*kF* (sub-features)
1	2	0.8694	10.9880	0.3200	2.7302	3, 21
2	3	1.2890	11.3440	2.7700	2.8956	3, 21, 27
3	4	1.7072	16.3530	4.7039	3.2544	3, 9, 21, 27
4	5	2.1224	68.8930	5.8300	5.3410	3, 6, 9, 21, 27
5	6	2.5366	490.705	10.4450	9.1431	3, 6, 9, 20, 21, 27

k-MPF4, and *k*-MPF5 algorithms on auto data on different *k*-values (Tables 11, 12).

Car data: The *Pf*, *Bf*, and *PEf* values of the Car data are given in Table 10. Here, the fourth sub-feature is the least promising feature and first and second sub-features are the highly promising features. We have applied all the algorithms on car data for different *k*-values. Tables 13 and 14 demonstrate the results.

Table 11 (Auto Data) and Table 13 (Car Data) demonstrate the execution time differences between *k*-MPF1 and *k*-MPF2 algorithms. The results show that the proposed *k*-MPF2 outperformed on all the cases. Similarly, Table 12 (Auto Data) and Table 14 (Car Data) show the execution time difference between *k*-MPF3, *k*-MPF4, and *k*-MPF5 algorithms and the results show that our proposed algorithm (*k*-MPF5) is the best in almost all the cases (except when *k* is 2, of Table 12 and when *k* is 5 of Table 14). The last column of all the tables (Tables 11, 12, 13, 14) displays the *kF* set for different *k* values with maximum *EkF* value [third column of all the tables (Tables 11, 12, 13, 14)].

Experiments on synthetic data: Tables 15, 16, 17 and 18 show the results of SYN1 and SYN2 data using all the algorithms. Our proposed algorithms (*k*-MPF2, *k*-MPF4, and *k*-MPF5) show the effectiveness in all the cases. Next, we discuss the scalability of all the algorithms on (a) *varying dimensions* (*d*-sub-features): Fig. 4 demonstrates the

Table 19 Comparison of *k*-MPF5, *k*-MPFb, and *k*-MPFsoc on auto and car datasets with different *k* values

k	Method	Auto			Car		
		Time	EkF	*kF*	Time	EkF	*kF*
2	*k*-MPF5	0.2804	1.2843	2, 14	0.298	0.6502	1, 14
	k-MPFb	0.6364	1.2843	2, 14	10.3147	0.9436	2, 18
	k-MPFsoc	0.5625	1.2843	2, 14	10.1075	0.6029	1, 15
3	*k*-MPF5	0.3010	1.8602	2, 6, 14	0.3001	0.9435	1, 14, 18
	k-MPFb	0.9362	1.8602	2, 6, 14	10.6392	0.9436	2, 15, 18
	k-MPFsoc	0.5630	1.8602	2, 6, 14	10.1223	0.8440	1, 5, 12
4	*k*-MPF5	0.4724	2.4268	2, 6, 8, 14	0.8965	1.2359	1, 14, 18, 21
	k-MPFb	1.2211	2.4268	2, 6, 8, 14	11.923	1.2359	2, 15, 18, 21
	k-MPFsoc	0.5635	2.3892	2, 6, 11, 14	10.1678	1.1326	1, 5, 12, 13
5	*k*-MPF5	0.4930	2.9559	2, 6, 8, 11, 14	2.9111	1.4817	1, 5, 14, 18, 21
	k-MPFb	1.3269	2.9559	2, 6, 8, 11, 14	12.629	1.4817	2, 7, 15, 18, 21
	k-MPFsoc	0.5644	2.8971	2, 3, 6, 11, 14	10.175	1.4230	1, 5, 12, 13, 16
6	*k*-MPF5	0.9702	3.4638	2, 3, 6, 8, 11, 14	7.09	1.7228	1, 5, 12, 14, 18, 21
	k-MPFb	1.4932	3.4638	2, 3, 6, 8, 11, 14	12.936	1.7228	2, 7, 12, 15, 18, 21
	k-MPFsoc	1.2238	3.4638	2, 3, 6, 8, 11, 14	10.6432	1.7114	1, 5, 12, 13, 16, 19

Table 20 Comparison of *k*-MPF5, *k*-MPFb, and *k*-MPFsoc on SYN1 and SYN2 datasets with different *k* values

k	Method	SYN1			SYN2		
		Time	EkF	*kF*	Time	EkF	*kF*
2	*k*-MPF5	0.7302	0.8902	4, 16	2.7302	0.8694	3, 21
	k-MPFb	3.597	0.8809	4, 6	10.212	0.8694	3, 21
	k-MPFsoc	3.0596	0.8809	4, 6	12.0946	0.8594	4, 21
3	*k*-MPF5	1.8956	1.3219	4, 6, 16	2.8956	1.2890	3, 21, 27
	k-MPFb	3.893	1.3061	3, 4, 6	10.923	1.2890	3, 21, 27
	k-MPFsoc	3.4706	1.2821	4, 6, 19	13.7337	1.2777	9, 14, 21
4	*k*-MPF5	2.2544	1.7535	4, 6, 11, 16	3.2544	1.7072	3, 9, 21, 27
	k-MPFb	3.969	1.7377	3, 4, 6, 11	11.031	1.7072	3, 9, 21, 27
	k-MPFsoc	3.4918	1.6751	4, 6, 9, 19	13.8538	1.7022	3, 9, 14, 21
5	*k*-MPF5	5.001	2.1720	4, 6, 11, 13, 16	5.341	2.1224	3, 6, 9, 21, 27
	k-MPFb	4.023	2.1619	3, 4, 6, 11, 16	12.321	2.1184	3, 9, 21, 22, 27
	k-MPFsoc	3.566	2.0742	2, 4, 6, 9, 19	13.900	2.1003	3, 9, 14, 18, 21
6	*k*-MPF5	8.1991	2.5710	2, 4, 6, 11, 13, 16	9.1431	2.5366	3, 6, 9, 20, 21, 27
	k-MPFb	4.932	2.5804	3, 4, 6, 11, 17, 18	13.936	2.5335	3, 6, 9, 21, 22, 27
	k-MPFsoc	3.999	2.4859	2, 4, 6, 9, 17, 19	14.1617	2.4972	3, 9, 14, 18, 21, 34

Table 21 Comparison between mPEP and PNPn values to show the effectiveness of *k*-MPF

	Auto ($k = 6$)		Car ($k = 6$)		SYN1 ($k = 7$)		SYN2 ($k = 11$)	
	mPEP	PNPn	mPEP	PNPn	mPEP	PNPn	mPEP	PNPn
k-MPF5	11.400	15	12.486	17.000	10.222	11	20.187	21.000
	Time: 0.9702		Time: 7.09		Time: 8.1991		Time: 12.23	
k-MPFb	11.400	15	12.486	10.917	10.222	11	20.187	19.991
	Time: 1.4932		Time: 12.936		Time: 9.83		Time: 24.00	
k-MPFsoc	11.400	15	12.486	15.000	10.222	11	20.187	20.000
	Time: 1.2238		Time: 10.6432		Time: 8.5423		Time: 23.9761	

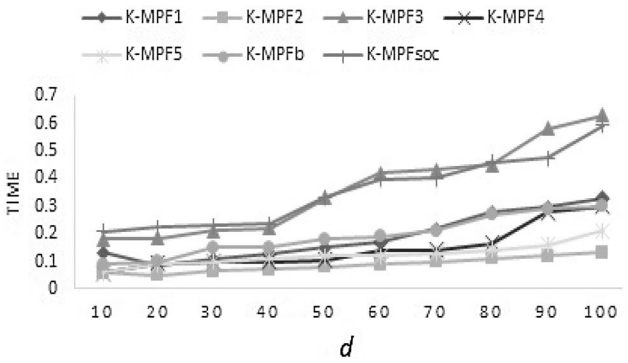

Fig. 4 Execution time on varying *d*

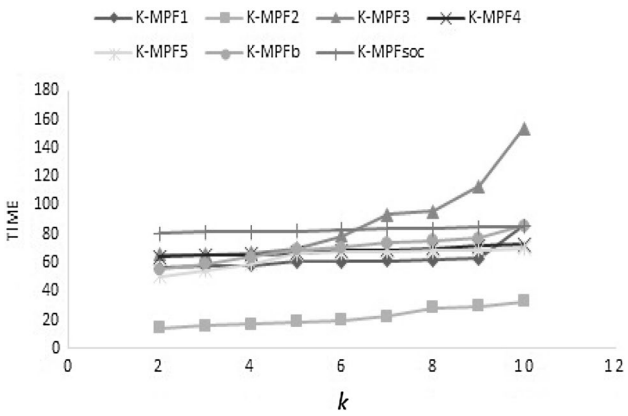

Fig. 5 Execution time on varying *k*

execution time of all the algorithms of varying number of dimensions (*d* sub-features). The results show that the execution time of our proposed *k*-MPF2 and *k*-MPF5 algorithms appears nearly flat when the number of sub-features (*d*) increases. But, the time complexities of the *k*-MPF3 and *k*-MPFsoc are not linear. (b) *Execution time on varying k-values*: The execution time of all the algorithms by varying the number of *k* on fixed data size is shown in Fig. 5. The graph indicates that the execution times of the proposed algorithms (*k*-MPF2, *k*-MPF4 and *k*-MPF5) are nearly flat when the number of *k* values increases. It is clear that the

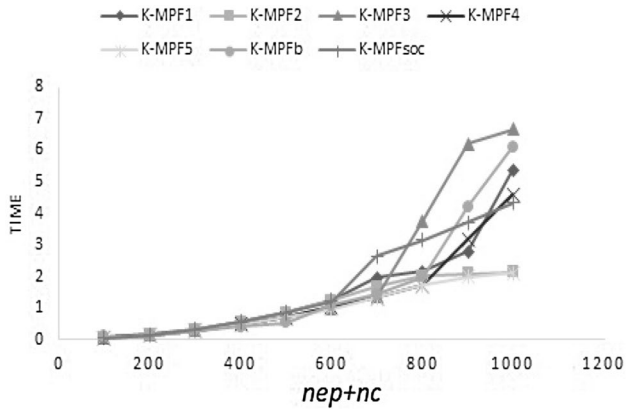

Fig. 6 Execution time on varying ($nep + nc$)

value of *k* has little effect on the execution time of our proposed algorithms. But it is not true for exhaustive versions of the *k*-MPF algorithms (*k*-MPF1 and *k*-MPF3). Here, execution efficiencies are not linear. (c) *Execution time on varying row* ($nep + nc$) *size*: Fig. 6 displays the execution time of the algorithms on different number of rows, i.e., $nep + nc$ (with fixed number of *k* and sub-features *d*). The results show that the execution time of exhaustive versions of the algorithms and existing algorithms take much time as compared to our proposed algorithms, when number of rows increases.

Comparison with existing techniques: Table 19 shows the comparison of *k*-MPF5, *k*-MPFb, and *k*-MPFsoc techniques using Auto and Car datasets with different *k* values. The results show that in most of the cases *k*-MPF5 needs the least time with respect to other two (*k*-MPFb and *k*-MPFsoc) techniques. The *EkF* values of *k*-MPF5 are the highest in all the time. Similarly, in Table 20, using two synthetic datasets (SYN1 and SYN2), the comparative study of the *k*-MPF5, *k*-MPFb, and *k*-MPFsoc techniques is presented. For SYN1 data, *k*-MPFsoc needs the least time for all the *k* values, but the *EkF* values of the *k*-MPF5 are the highest in all the cases. In SYN2, the *k*-MPF5 needs the least time and the *EkF* values are the highest for all the *k* values.

Effectiveness of the k-MPF algorithms: Table 21 exhibits the effectiveness of the *k*-MPF5, *k*-MPFb, and *k*-MPFsoc algorithms on different datasets and *k* values. It is observed

that *k*-MPF5 is better than *k*-MPFb, and *k*-MPFsoc in terms of time and PNPn in almost all the cases.

5 Conclusion

In this paper, we formulated *k*-most promising features (*k*-MPF) with compatibility and non-compatibility issues. Three algorithms *k*-MPF2, *k*-MPF4, and *k*-MPF5 are proposed to identify the *kF* features to develop new competitive products efficiently to set the blue ocean strategy. We have compared our proposed algorithms with exhaustive versions of the algorithms and existing algorithm using real and synthetic datasets, and the results demonstrate that the proposed algorithm is the best in all most all the cases. We can use our proposed algorithms in business decision to set the blue ocean strategy. The proposed algorithms can identify the *k*-most promising features along with the highly promising, least promising, and basic features.

Finally, in this paper, the issues we have handled are novel and important to the area of a business decision, and we observed that our evaluation parameters have little evidence that these synthetically generated "new" products would be truly popular in the market and we do not attempt to exhibit the practical application value, hence overcoming such limitation of the paper is subject to future work.

Acknowledgements We are grateful to the anonymous reviewers for their constructive comments on this paper. Moreover, we give our heartfelt thanks to Dr. Srabani Mukhopadhyaya and Dr. Pallavi Singh for their unconditional help.

References

1. Kotler P, Silva GD, Armstrong G, Haque E (2013) Principles of marketing: a south Asian perspective, 13th edn. Pearson, New Delhi
2. Sauter VL (2008) Competitive intelligence systems: handbook on decision support systems 2: variations. Springer, Berlin, pp 195–210. ISBN: 978-3-540-48716-6
3. Hohhof B (1994) Developing information systems for competitive intelligence support. Library Trends 43(2):226–238
4. Islam MS, Liu C (2016) Know your customer: computing k-most promising products for targeted marketing. VLDB J 25(4):545–570
5. Kim WC, Mauborgne R (2005) Blue ocean strategy: how to create uncontested market space and make the competition irrelevant. Harvard Business School Press, Boston ISBN: 978-1591396192
6. Miah M, Das G, Hristidis V, Mannila H (2008) Standing out in a crowd: selecting attributes for maximum visibility. In: Proceedings of the 24th international conference on data engineering, pp 356–365
7. Tan P-N, Steinbach M, Kumar V (2009) Introduction to data mining. Pearson Education Inc, New Delhi
8. Lin CY, Koh JL, Chen ALP (2013) Determining k-most demanding products with maximum expected number of total customers. IEEE Trans Knowl Data Eng 25(8):1732–1747
9. Goulding IC (1983) New product development: a literature review. Eur J Mark 17(3):3–30
10. Wan Q, Wong RCW, Ilyas IF, Ozsu MT, Peng Y (2009) Creating competitive products. In: Proceedings of the 35th international conference on very large data bases, pp 898–909
11. Wong RC-W, Fu AW-C, Pei J, Ho YS, Wong T, Liu Y (2008) Efficient skyline querying with variable user preferences on nominal attributes. Proc VLDB 1(1):1032–1043
12. Zhang Z, Lakshmanan LVS, Tung AKH (2009) On domination game analysis for microeconomic data mining. ACM Trans Knowl Discov Data 2(4):18–44
13. Michalek JJ, Feinberg FM, Papalambros PY (2004) An optimal marketing and engineering design model for product development using analytical target cascading. In: Proceedings of the TMCE
14. Blijlevens J, Creusen MEH, Schoormans JPL (2009) How consumers perceive product appearance: the identification of three product appearance attributes. Int J Des 3(3):27–35
15. Wu T, Xin D, Mei Q, Han J (2009) Promotion analysis in multi dimensional space. In: Proceedings of the 35th international conference on very large data bases, vol 2, issue 1. pp 109–120
16. Koh J-L, Lin C-Y, Chen ALP (2014) Finding k most favorite products based on reverse top-t queries. VLDB J 23(4):541–564
17. Peng Y, Wong RC, Wan Q (2012) Finding top-k preferable products. IEEE Trans Knowl Data Eng 24(10):1774–1788
18. Dellis E, Seeger B (2007) Efficient computation of reverse skyline queries. In: Proceedings of the 33rd international conference on very large data bases, pp 291–302
19. Lian X, Chen L (2008) Monochromatic and bichromatic reverse skyline search over uncertain databases. In: Proceedings of the 27th ACM SIGMOD international conference on management of data, pp 213–226
20. Wu W, Yang F, Chan CY, Tan KL (2008) FINCH: evaluating reverse k-nearest-neighbor queries on location data. In: Proceedings of the 34th international conference on very large data bases, pp 1056–1067
21. Leisch F, Weingessel A, Hornik K (1998) On the generation of correlated artificial binary data. Working paper series, SFB "adaptive information systems and modelling in economics and management science". Vienna University of Economics. http://www.wu-wien.ac.at/am