Knowledge-Based Systems 164 (2019) 292–308

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Personalized service selection using Conditional Preference Networks

Hongbing Wang [a,*], Yong Tao [a], Qi Yu [b], Hong Tianjing [a], Chen Xin [a], Wu Qin [a]

[a] School of Computer Science and Engineering, Southeast University, SIPAILOU 2, NanJing 210096, China
[b] College of Computing and Information Sciences, Rochester Institute of Tech, USA

ARTICLE INFO

Article history:
Received 1 April 2018
Received in revised form 27 October 2018
Accepted 29 October 2018
Available online 15 November 2018

Keywords:
CP-net
Dimensionality reduction
Top-*k* query

ABSTRACT

Top-*k* and skyline techniques have been used to address preference based queries for effective service selection. However, they do not consider the dependencies between attributes in user preferences. In this paper, we focus on developing top-*k* indexing methods based on Conditional Preference Networks. We first determine whether the correlation among service attributes is clear and definite. After that, we employ dimensionality reduction to reduce the dimensionality of the service space. We then use top-*k* query to further improve the scalability. We conduct extensive experiment and compare with other competitive indexing mechanism to demonstrate the effectiveness of the proposed approach.

1. Introduction

With the large number of services being developed and deployed nowadays, Web service recommendation techniques have gained great popularity which can provide personalized services to users based on their preferences. There are many models used to describe user preferences. Top-*k* [1–4] and Skyline [5–9] are the two representative preference models for quantitative preference processing. Top-*k* retrieval assumes a utility function, which can be used to determine the object scores calculated based on the value of each attribute. In contrast, skyline does not depend on any specific utility function. Instead, it uses Pareto-dominance to decide object priority. However, both models are primarily designed for numerical attributes. They do not consider the relationship between attributes, either. Conditional Preference Network (CP-net) is another widely used preference model that provides a good tradeoff between expressiveness and simplicity [10–13]. Using a set of attributes we obtained from candidate objects on a CP-net, we can get users' preference of each attribute and independencies thereof. The nature of the CP-net model has been extensively studied in academia [11,14,15], but few methods have been developed for data/service retrieval based on CP-net. One exception is in [16], which employs CP-net for Top-k retrieval. However, it lacks an efficient indexing method for multidimensional data, leading to poor performance.

The difference between traditional indexing methods and dimensionality reduction methods is that the former assumes each dimension is equally important. But for the data described by CP-net, this assumption may not be true. When designing an indexing method, the relevance of attributes needs to be considered. Based on this, the dimensionality reduction indexing method can be used to build the index to satisfy the user preference model. In particular, we will develop an efficient indexing mechanism that combines dimensionality with Top-k query. We use CP-net to model user's preference. We then evaluate whether the correlation between each dimension of the data registration center is determined by the multidimensional user preference model. If so, we exploit Hilbert Curve [17] to reduce the dimensionality to one. Otherwise, we apply Principal Component Analysis (PCA) [18] for dimensionality reduction. Last, Top-k query is applied to reduced space for efficient service retrieval.

Our main contributions presented in this paper are the following: (1) We proposed a method composed of Top-k retrieval algorithm and indexing mechanism based on CP-net, which helps us to select a more suitable and personalized service for the user. (2) We also present several data dimensionality reduction approaches to improve the efficiency of data retrieval while meeting high-dimensional data CP-net. (3) We conduct a lot of experiments to compare our method with others, and the results show that the method we put forward is effective and feasible.

The rest of this paper is organized as follows. In Section 2, we discuss related work. We review the CP-net model and its primary properties in Section 3. In Section 4, we introduce the Hilbert Curve and Principal value analysis (PCA), specifying their dimensionality reduction process. Section 5 presents the Top-*k* indexing scheme based on CP-net. In Section 6, we state our experimental setup and comparative analysis of the empirical result. Finally, conclusion and future work are given in Section 7.

2. Related work

Incorporation of user preference in query processing has received increasing interest in recent years. The primary problem is

[*] Corresponding author.
E-mail addresses: hbw@seu.edu.cn (H. Wang), qi.yu@rit.edu (Q. Yu).

https://doi.org/10.1016/j.knosys.2018.10.040
0950-7051/© 2018 Elsevier B.V. All rights reserved.

how to model user preferences and how to integrate them into the database query language. In [19], Wenzel et al. proposed an integrated database-driven recommendation approach using online social networks to find people with common interests. In [20], Sarwat et al. conducted extensive experiments that study the performance of personalized recommendation applications based on an actual implementation using real Movie recommendation and location-aware recommendation scenarios. The model proposed in [21] provides a set of constructors to express basic and atomic user preferences. More complex preferences can be expressed by compositions of the constructors. In [22] and [23], Yan et al. and Hsueh et al. used the skyline operator to pre-process services and find a set of candidate services which can satisfy users' requirements. In [24], Benouaret et al. proposed an approach to select Top-k cloud services combining the trust, determined by the reputation of the provider, and the QoS. In [25] Purohit and Kumar improved the PROMETHEE method and applied it to the most eligible web services. A Maximizing Deviation based hybrid weight evaluation mechanism is adopted to select the Top-k web services matching closely with the QoS requirements of the users. In [26], Wang et al. present a new user interaction model in multi-objective query processing which allows the users to preset Weight Profiles and their logical descriptions. Weight Profiles contain objective preferences for the users before the query is executed. The work in [27,28] considers how to enable preference in a relational database. It proposes Preference SQL for expressing user preferences and defines rules for translating Preference SQL into traditional queries of relational databases. A contextual preference model is proposed in [29] that is closest to the idea of CP-net. In [30], the author adopts the constrained CP-net to model a set of constraints and preferences for expressing users' preferences, which return a set of outcomes in the form of a suggestion list. This list is then sorted according to user preferences. However, these models do not consider the dominance relationship, but use quantitative measures to rank query results. In [31,32], Boubekeur et al. proposed to use CP-net to make their flexible information retrieval method more easy and clear to represent qualitative queries. The author mainly focuses on document indexing and query evaluation based on the CP-net theoretical foundations. The semantics of CP-net have been applied to the relational database in [33,34]. However, all of them do not address the problem of Top-k retrieval. An approach in [35] presented Top-k retrieval using CP-net, which can efficiently retrieve the most preferred data items based on a user's CP-net. The approach comprises a Top-k retrieval algorithm and an indexing mechanism. Although the proposed method proves to be effective in some degree, it ignores the problem of high dimensional data, and does not use dimensionality reduction methods.

For a high-dimensional data CP-net, dimensionality reduction can help reduce the data retrieval challenges. There are many dimensionality reduction approaches, which are mainly divided into two categories. One is linear dimensionality reduction with PCA [36] and Wavelet Transform [37] as the representatives. Harrou [38] propose a strong PCA dimension reduction method, which learns to ignore a large part of detailed spatial structure of input and thereby estimates a linear pooling matrix. The other is nonlinear. PCA transforms multiple variables into fewer dimensions, while wavelet transform is mainly used to extract local features. Hilbert curve [17] is a mapping method between multi-dimensional space and 1-dimensional space that is mostly used in the field of image processing and multi-dimensional data indexing. For example, in [39], the author proposes to use the index-based Hilbert-Temporal Join algorithm that maps multidimensional temporal data into a Hilbert curve space. As for the Hilbert curve code generation, the most important step of Hilbert curve dimension reduction, there are two methods: a table-driven method with high computational complexity and a calculation method. Table-driven approach generates a curve by scanning the code scan list. Fish [40] gives an iterative version for one-dimensional to two-dimensional mapping. Cole [41] provides a reverse version for two-dimensional to one-dimensional mapping. Jin and Mellor-Crummey [42] propose a framework to efficiently generate space-filling curve. [43] puts forward a new algorithm for N-dimensional Hilbert scanning. The method of calculation calculates the one-to-one mapping. Butz [44] calculates the corresponding coordinates of an arbitrary point on the curve. [45] suggests some practical improvements to the algorithm proposed by Butz. The Faloutsos Roseman gave a non-iterative method to achieve this mapping by analyzing the relation between Z-order and Hilbert. Both of these two methods are widely used in Artificial intelligence, such as in [46], the author uses multiscale PCA-learned filters for dynamic texture recognition, and in [47], the author use Hilbert transform to evaluate the error of phase estimation and to guide optimal filter design.

Although there are many dimensionality reduction methods, few of them are used to retrieve data in conjunction with CP-net, which will be used in our work to handle the dependencies among attributes. We propose to apply the Hilbert Curve and Principal Component Analysis for dimensionality reduction to process the CP-net data for efficient retrieval. We then use depth-first Top-k retrieval to search the satisfied services.

3. CP-net model

CP-net [11] is a graphical model for representing and reasoning with conditional preference in a compact, intuitive and structural manner including directed dependence graph (DDG) and conditional preference tables (CPTs) [10]. Here, we describe the CP-net model and its main properties.

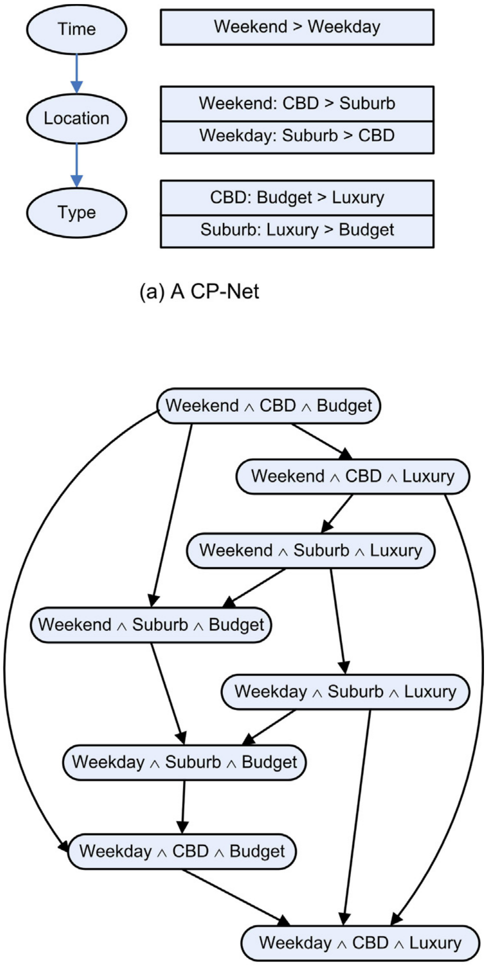
3.1. Model definition

The model of CP-net is defined as follows.

Definition 1 (*CP-net*)**.** Let $V = \{X_1, \ldots, X_n\}$ be a set of attributes. A CP-net over V is a directed graph G over $X_1, \ldots, X_n$, whose nodes are annotated with *conditional preference tables*, denoted by $CPT(X_i)$ for each $X_i \in V$. Each conditional preference table $CPT(X_i)$ associates a total order of X_i's values with each instantiation of X_i's parents.

Fig. 1 provides an illustrative example on CP-net. Suppose that Anny plans to travel to Sydney. She has many things to decide, such as when to leave and which hotel to stay. We can see that the CPT provides important information. For example, spending a weekend in Sydney is more attractive than a weekday for her. She is willing to stay in the suburb if it is a working day but she prefers the CBD (Central Business District) if it is a weekend. In addition, what kind of hotel she will stay depends on where she lives. In the CBD, she prefers economic hotel; in the suburb, she prefers a luxurious hotel. From the figure, we can understand that an attribute may determine the value of another attribute and a CP-net can express this kind of conditional preference. Based on a CP-net, we could do the dominance testing, which is used to compare two attributes to select the dominant one.

Definition 2 (*Dominance*)**.** Let N be a CP-net over a set of attributes V. Given two instances e_1 and e_2, if for every attribute $X_i \in V$, (1) e_1 and e_2 assign the same values to X_i's parents, and (2) based on the corresponding entry in $CPT(X_i)$, e_1 assigns an equal or better value to X_i than that assigned e_2, then we say that e_1 *dominates* e_2. We denote it by $N \models e_1 \succ e_2$. Moreover, the dominance relationship is transitive, i.e. $N \models e_1 \succ e_2 \wedge N \models e_2 \succ e_3$ implies $N \models e_1 \succ e_3$.



(a) A CP-Net

(b) Induced Preference Graph

Fig. 1. CP-net example.

According to above definition, we can conclude that instantiation {(time = weekend) ∧ (location = CBD) ∧ (type = budget)} dominates {(time = weekend) ∧ (location = suburb) ∧ (type = luxury)} based on Anny's CP-net in Fig. 1(a). By specifying the dominance relationships among all the possible instances, we can get the detailed preference graph of Anny, which is shown in Fig. 1(b). It should be noted that we consider only acyclic CP-net in this paper as a cyclic CP-net can beget conflicts in practice.

3.2. The properties of CP-net

We present some of the important properties of CP-net [11,14,15] in this section.

Property 1 (*Dominance Testing is Complex*). *Given a CP-net N and two instances e_1 and e_2, the dominance testing aims to determine if one of the instances dominate the other. The testing must reach one of the following three conclusions – (1) $N \models e_1 \succ e_2$ or (2) $N \models e_2 \succ e_1$ or (3) $N \nvDash e_1 \succ e_2 \wedge N \nvDash e_2 \succ e_1$ (also known as* indifference*).*

Property 2 (*Outcome Optimization is Easy*). *Given a acyclic CP-net N, outcome optimization aims to find the best possible instance that cannot be dominated by any other possible instance.*

Property 3 (*Ordering Testing is Easy*). *Given a acyclic CP-net N and two instances e_1 and e_2, the ordering testing aims to reach one of the following conclusions – (1) $N \nvDash e_1 \succ e_2$ or (2) $N \nvDash e_2 \succ e_1$ or (3) both.*

For Property 1, the solution of dominance testing depends on the size of the CP-net. Boutilier et al. [11] have proven that dominance testing for a binary-valued acyclic CP-net is NP-complete. Although some special CP-net, such as tree structure, can find effective algorithms to solve the problem, in general, such dominance tests are expensive. Therefore, we conclude that dominance testing for CP-net is intractable in general. For Property 2, it implies that outcome optimization can be solved in linear time. A simple algorithm for optimizing the result is forward scanning. This algorithm scans from the top to the bottom along the CP-net, that is, from the parent to the child, and gives the best value to the child node according to the value of the parent node. As a result, although optimization can be easily solved, it is not so useful in the real world, which is attributed to the fact that the best cases usually do not exist. In order to find a real optimal instance, one way is to check a possible instance one after another, according to the detailed preference map (Fig. 1(b)), until a certain instance is found. For Property 3, it shows that ordering testing which aims to find Top-k instances that are not dominated by any other instance, can also be solved in linear time. Therefore, ordering testing is more feasible than dominance testing. In a Top-k retrieval ordering relation is very important. Although it cannot decide whether an instance dominates another instance, it can at least illustrate that an instance is not worse than another instance. And if we use the ordering relation to sort a set of instances, a higher rank instance will certainly be better than an instance with a lower rank in a different degree. As is shown later, ordering relationship is sufficient for Top-k retrieval. Algorithm 1 demonstrates the ordering testing. The complexity of the algorithm is $O(n)$, where n represents the size of the CP-net, i.e., the number of nodes plus the number of edges.

Algorithm 1: Ordering Testing

```
Input: Let $N$ be a CP-net over attributes $V$ and $e_1$ and $e_2$ be two
       instances.
begin
    foreach $X_i \in V$ do
        if $e_1$ and $e_2$ assign the same values to the parents of $X_i$ then
            if given $e_1$'s and $e_2$'s values on the parents of $X_i$, $e_2$ assigns
            a more preferred value to $X_i$ than $e_1$ does then
                set $N \nvDash e_1 \succ e_2$ to true;
            else
                set $N \nvDash e_2 \succ e_1$ to true;
```

4. Data dimensionality reduction

For a high-dimensional preference data described by the CP-net, we utilize dimensionality reduction methods to decrease the query space to effectively accelerate the retrieval. Dimensionality reduction methods include linear and nonlinear dimension reduction techniques. The advantages of linear dimensionality reduction technique are that it is simple and intuitive, and has no local extremum or relative effectiveness, which is easy to implement. In general, nonlinear dimensionality reduction methods are based on linear dimensionality reduction methods to expand nonlinear characteristics or use neural network to optimize dimensionality reduction [48]. Many techniques for dimensionality reduction have been proposed in the literature. In [49], PCA is by far one of the most popular algorithms for dimensionality reduction. The PCA, especially, is a well-known technique, whose idea is simple and easy to understand. Additionally , the algorithm of PCA is brief and efficient. However, non-linear dimensionality reduction methods suffer from huge computational costs [50]. In this paper, we use the dimensionality reduction method to reduce the dimension of the data and then establish the index, in order to find the index which can improve the efficiency of Top-k retrieval. Although the data in CP-net model is nonlinear, we do not need to consider the situation that the nonlinear characteristics of data sets may be lost after reducing the dimensions. Compared with the complex nonlinear dimensionality reduction method, the linear dimensionality reduction is sufficient to reduce the data complexity and identify the most important features. Therefore, in this paper, we use linear dimensionality reduction method.

By analyzing the relationship between data attributes, we use different dimensionality reduction for different data sets modeled using a CP-net. The process is detailed as follows. By using the Top-k search engine in user preference model to retrieve services in the registration center in a database, the retrieval result contains at least two service information. If the service information in search result is not the same, it means different user preferences on each attribute is not the same. Hence, the correlation of dimension is determined. If the service information in the retrieval result contains the same part, it means the correlation between the data dimension is uncertain.

Based on whether the correlation is certain, we can use the PCA and Hilbert Curve to reduce the data dimensionality. When the correlation of attributes in a CP-net is determined, there is no need to further deal with the dependencies between data attributes. And in this case, Hilbert Curve can be used to process the data sets, which does not affect the information of the original data set, but only maps the high-dimensional data to the one-dimensional data set space. In this paper, we mainly uses Hilbert curve to reduce the dimensionality, which is used to map a large number of data into one dimensional data space. And then we establish corresponding indexes, which contributes to reducing the number of single data attribute comparison times and improving the retrieval efficiency. When the correlation of CP-net data attributes is uncertain, the data set cannot be processed directly. In this case, PCA can be used to determine the correlation between data attributes and further deal with data optimization. PCA is able to analyze the main affecting factors from multiple things, thus revealing the essence of these things. In fact, if the main attributes of the things are reflected in several main variables, we just need to separate these variables and analyze them in detail. In this paper, PCA will used to extract the main characteristics of a data set to satisfy the user's preference and achieve the requirements of dimensionality reduction. In summary, we apply Hilbert Curve to deal with dimensionality reduction in the situation that the correlation between dimensions is clear and definite and use PCA otherwise.

4.1. Hilbert curve

Hilbert Curve [17] describes a one-to-one mapping between N-dimensional space and one-dimensional (1-D) space, which has an important position in the field of image processing and multi-dimensional data indexing. In this paper, we can obtain a unique result by Hilbert dimensionality reduction by maintaining the original partial order, so as to achieve maximum and lossless dimensionality reduction. Fig. 2 is the Hilbert Curve dimensionality reduction process flowchart based on a table-driven method.

Hilbert Curve is usually used to simulate the growth of organisms. Let R^N denote a N-dimensions space. $V_N, \dots, V_2, V_1$ is used to represent the coordinate value of each dimension of R^N. In order to facilitate computation, every coordinate is encoded in binary. We define if a Hilbert Curve can fill a $2^m \times 2^m \times \cdots \times 2^m \times (2^{mN})$ N-dimensions hypercube space. This curve can be regarded as the $m_t h$ generation of N-dimensional Hilbert Curve and denoted by H_m^N. The coordinate of R^N transferred by Hilbert Curve order is called the Hilbert code, which is denoted by H-order.

In the description of the Hilbert Curve, we can interpret this $2^m \times 2^m \times \cdots \times 2^m \times (2^{mN})$ hypercube space in two different ways. The first one is that it is a N-dimensional unit cube, which is divided into $2^m \times 2^m \times \cdots \times 2^m \times (2^{mN})$; the second one is a N-dimensional hypercube space with 2^m length for each side. For the latter case, in the code mapping, a N-dimensional Hilbert Curve is a N-dimensional Hilbert unit when $m = 1$ and is expressed as C^N. A N-dimensional Hilbert gene is a series of information list for transformation of coordinates. It controls how H_m^N generates H_{m+1}^N, which is denoted by G^N.

In short, multi-dimensional coordinates are translated into binary code representing multi-dimensional data. Then based on the new binary value, we query the C^N table to get the H-order. We then place the binary bit converted from H-order the highest position on the result. Finally, we obtain the conversion instructions based on the H-order on the G^N table.

4.2. Principal component analysis

PCA [18] is a linear dimensionality reduction method that transforms multiple variables into a few synthetical variables based on the internal structure of the covariance matrix of original variables. For its simplicity and optimal linear reconstruction error, we adopt it to process CP-Net data dimensionality reduction. Algorithm 2 shows the PCA dimensionality reduction process.

Algorithm 2: Principle Component Analysis

```
Parameter Specification: $X$ is a $n_1 * n_2$ matrix while $n_1$ is the
sample size and $n_2$ is the number of variables. $N$ indicates that it
is projected into the space of $N$ coordinates. $P$, as the return value,
is a transformation matrix. $D$ represents all the eigenvalues and $R$
is the result of projection.
$Y = X_T$;
foreach row of $Y$ do
    foreach value of this row do
        Compute the mean value of these values in the row;
    All the values in each row of $Y$ minus the mean value of this
    row;
$C = Y * Y_T$;
Compute the eigenvalues and eigenvectors of $C$;
Sort the eigenvalues and eigenvectors of $C$ in descending order;
Combine the ordered eigenvectors into a transformation matrix;
Calculate the sum of the first $K$ eigenvalues and divide it by the
sum of all the eigenvalues, so as to get the contribution rate of the
first $K$ dimensions;
$R = X * P$;
```

For N-dimensional data, we first calculate the mean value of the original data and subtract it. Then we calculate the sample covariance matrix. Afterwards, we calculate the eigenvalue and eigenvectors to obtain independent variables. We sort the eigenvectors according to their corresponding eigenvalues. The first eigenvector with the largest eigenvalue is called the first principal component and so on. In this way, by choosing the set of eigenvectors with the largest eigenvalues, we construct a new variable space that keeps the most information (i.e., variance) of the original data space.

5. Top-k indexing based on CP-Net

Top-k retrieval is to identify the k most preferred objects. This section presents several indexing methods based on the depth-first algorithm.

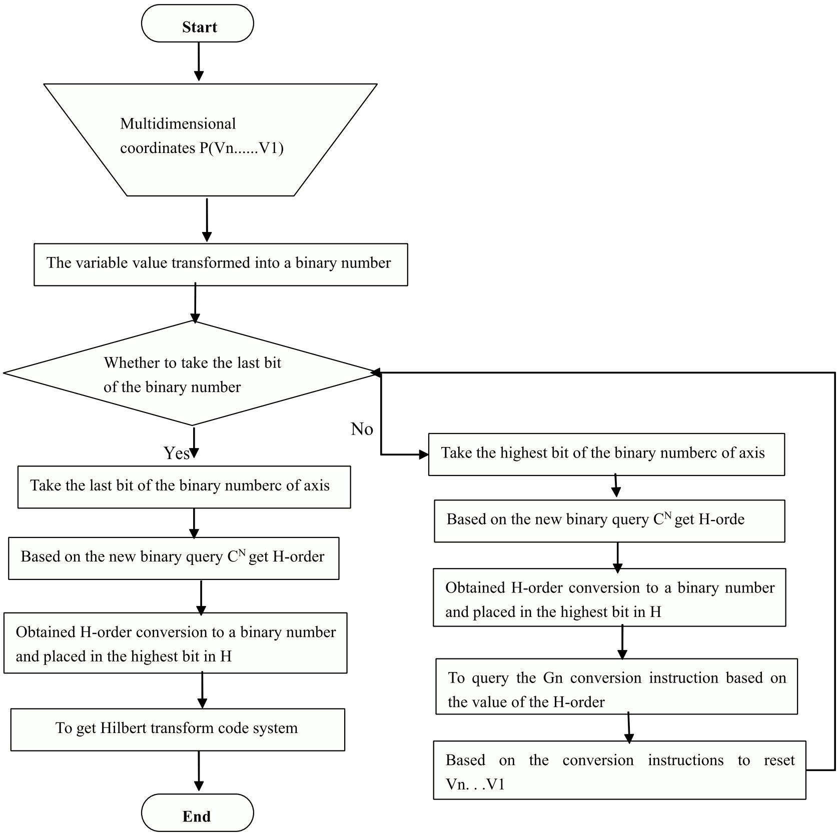

Start
Multidimensional coordinates P(Vn......V1)
The variable value transformed into a binary number
Whether to take the last bit of the binary number
Yes
No
Take the last bit of the binary numberc of axis
Based on the new binary query C^N get H-order
Obtained H-order conversion to a binary number and placed in the highest bit in H
To get Hilbert transform code system
End
Take the highest bit of the binary numberc of axis
Based on the new binary query C^N get H-orde
Obtained H-order conversion to a binary number and placed in the highest bit in H
To query the Gn conversion instruction based on the value of the H-order
Based on the conversion instructions to reset Vn. . .V1

Fig. 2. Hilbert flow chart.

Definition 3 (*Dependency Order of Attribute*)**.** Given a CP-Net N and let $V = \{X_1, ..., X_n\}$ be a set of attributes, If in the CP-net N, X_i is pointed by X_j, it means preference for X_i depends on X_j. We regard X_j as the parent of X_i, denoting X_j as $P(X_i)$, namely there is a dependency order from X_j to X_i.

Definition 4 (*Top-k for CP-Net*)**.** Given a CP-Net N and a set of instances E, the top-k instances of E on N is a set of k instances $E' = \{e_1, \ldots, e_k\} \subset E$, such that no instance in $E - E'$ dominates any instance in E'.

From the conclusion of Property 1, we know that it is computationally expensive to retrieve Top-k objects with dominance testing. Fortunately, ordering dominance (Property 2) is enough to get a valid Top-k set. Here are two lemmas [16].

Lemma 1. *Given a CP-Net N and a set of instances E, we sort E based on the outcomes of the ordering testing, i.e., if $N \nvDash e_2 \succ e_1$, then e_1 is ranked before e_2. Then, the first k instances of the sorted list is a top-k set of E on N.*

Based on the Lemma 1, we propose to use the Depth First Top-k Retrieval to improve the efficiency of the selecting without scanning the whole data. This method only retrieves the Top-k instance. The pseudo-code of depth-first Top-k retrieval for CP-Net is given in Algorithm 4. The process of the algorithm is as follows: firstly, it sorts the attributes based on the dependency order in the CP-net. Secondly, based on the sorted order of result, this method chooses values from the k-optimal of the attributes. For each attribute X_i, it retrieves the list of its instantiations $v^i_m \succ v^i_{m-1} \succ \ldots \succ v^i_1$ from the CPT, and evaluates the instantiations one by one according to the preference order. If one instantiation can satisfy the data set, then this algorithm extends this instantiation by traversing forward to the next attribute in the sorted attribute list. Lastly, when all the instantiations of an attribute are evaluated, the algorithm traverses back to the previous attribute to continue its evaluation. When all the attributes are evaluated and a complete instantiation is obtained, then the algorithm outputs the corresponding data instance as a member of the Top-k results. This depth-first traversal continues until k instances are identified.

```
Algorithm 3: Satisfiability test
Input: let $N$ be a CP-net over attributes $V$,
       $I = \{X_1 = v^1, ..., X_i = v^i\}$ an item and
       $E = \{v^1_{m_1}, v^2_{m_2}, ..., v^n_{m_n}\}$ be a set of instances.
begin
    sort $V$ based on the top-down (ancestor-descendant) order of
    $N$, let the sorted list be $X_1, X_2, ...X_n$;
    let $CPT(X_1) = \{v^1_m \succ v^1_{m-1} \succ ... \succ v^1_1\}$;
    while $j < i$ do
        if the value of $v^j$ is the same as $v^j_{m_j}$ existing in $E$
        then $j++$;
        else
        return $I$ is not satisfiable;
        break;
        return $I$ is satisfiable;
```

Algorithm 3 shows the satisfiability test that will be used in Algorithm 4. The algorithm takes the item I that needs to be tested as the input. The algorithm proceeds by sorting the attribute nodes in advance. After that, it compares the first attribute value of the item with that in database (E). If a match is found, it extends the item to compare with its second attribute value. Otherwise, it changes to compare with the second value of the first attribute. If the values of an attribute cannot be matched by E, the item is not

Table 1
Instances in data set.

Attribute	Time	Location	Type
Data	Weekend	Suburb	Luxury
	Weekday	CBD	Budget
	Weekday	Suburb	Luxury

Algorithm 4: Depth First Top-*k* Retrieval for CP-Net

```
Input: let $N$ be a CP-net over attributes $V$, and $E$ be a set of
       instances.
begin
    sort $V$ based on the top-down (ancestor-descendant) order of
    $N$, let the sorted list be $X_1, X_2, ...X_n$;
    let $CPT(X_1) = \{v_m^1 \succ v_{m-1}^1 \succ ... \succ v_1^1\}$;
    for $j = 1...m$ do
        push $\{X_1 = v_j^1\}$ into the stack $ST$;
    while $ST$ is not empty do
        pop out an item from $ST$, let it be
        $I = \{X_1 = v^1, ..., X_i = v^i\}$;
        check if $I$ can be satisfied by $E$;
        if $I$ is satisfiable then
            if $i == n$ then
                /* $i == n$ means that $I$ is a complete instantiation. */
                output the instances in $E$ that satisfy $I$;
                if the number of the total output instances reaches k
                then
                    exit;
            else
                check the entry in $CPT(X_{i+1})$ that corresponds to
                $I = \{X_1 = v^1, ..., X_i = v^i\}$;
                let the entry be $v_m^{i+1} \succ v_{m-1}^{i+1} \succ ... \succ v_1^{i+1}$;
                for j=1...m do
                    push $\{X_1 = v^1, ..., X_i = v^i, X_{i+1} = v_j^{i+1}\}$ into $ST$;
```

satisfiable by E. In this way, the algorithm determines whether the item can be satisfied by E.

To better understand the satisfiability test algorithm, we use an example to further illustrate how it works. Suppose that the corresponding data set contains only 3 instances, as shown in Table 1. According to the CP-net in Fig. 1a, the algorithm first sorts the attribute nodes into time, location, and type. If the input item is given by $I = \{time = weekend, location = CBD, type = luxury\}$, we can clearly find that the first instance of the data set can be met with I. So we can conclude that item I can be satisfied by the data set.

Algorithm 4 gives the details of the depth first top-k retrieval for CP-Net, which consists of a series of satisfiability tests while avoiding to assess the instances that cannot be satisfied. The algorithm starts with those top attributes without parent nodes in the CP-net. Every time, the algorithm only considers instances with a subset of attributes. If there is an actual instance in the database that has the same value on the attribute, then the instance can be satisfied by the database. Next, the algorithm expands the instance according to the topological order and repeats these steps. If the database cannot meet the instance, other optimal values of the instances will be evaluated. To ensure that the whole process follows the depth priority, a stack ST is used to control the order of partial composition to be evaluated. When a complete instance of satisfaction is located, it is returned as one of the $Top - k$ results in the database.

Suppose that the corresponding data set contains only 3 instances, as shown in Table 1. According to the CP-net in Fig. 1a, Algorithm 4 firstly sorts the attribute nodes into time, location, and type. The first instance is $\{time = weekend\}$, which can be met by the first instance of the data set. Then, we extend the instance to be $\{time = weekend\}\{location = CBD\}$. However, the extended instance cannot be found in the data set. Therefore, the value of the location is changed to the second value in its preference sequence, that is, $\{time = weekend\}\{location = suburb\}$. Continuing like this, we can get the complete instance $\{time = weekend\}\{location = CBD\}\{type = luxury\}$ step by step, which can be satisfied in data set. Finally, the algorithm returns this instance as Top-1 result. In this way, we can finally find the Top-k instances satisfying E.

In this algorithm, we often need to test whether an instance of a set of attributes can be satisfied in the data set. If we take the satisfiability test of each part combination as a unit of calculation, then the worst case of computational complexity for Algorithm 4 is $O(D \times C)$, and the best case is $O(D)$, where D represents the number of attributes (dimensionality) and C represents the values (cardinality) of each attribute. If we find a complete instance of satisfaction, which meets every attribute's last value in the data set, the maximum possible computational cost to retrieve a single result is $O(D \times C)$. The best case corresponds to finding a complete instance that is composed of the first value of every attribute, leading to a minimum possible computational cost as $O(D)$. The above analysis does not consider the retrieval time from the database. Since the depth first searching of the outcome space constructs a searching tree, if all the nodes in the searching tree are tested, in the worst case, the time complexity of Algorithm 4 is $O(C^D)$. In addition, the size of E is a constant, which is determined by the database. Therefore, it does not affect the complexity of the algorithm.

Lemma 2. *Let N be a CP-Net over the attributes V. Let V' be a subset of V, i.e., $V' \subset V$, such that there is no $a \in V'$ whose parents belong to $V - V'$. Let N' be a subgraph of N which contains all the attributes in V' but not a single attribute in $V - V'$. Then,*

(1) N' is a valid CP-Net on V';

(2) Let e_1 and e_2 be two instantiations on V, and e_1' and e_2' be their mappings on V' respectively. If $e_1' \neq e_2'$ and $N' \nvDash e_1' \succ e_2'$, then $N \nvDash e_1 \succ e_2$.

The Lemma 2 can be proved as follows. For one thing, since all parents of node in V' are in V', then all the CPT of N' are complete. So N' is a valid CP-net of V'. For another thing, due to $e_1' \neq e_2'$ and $N' \nvDash e_1' \succ e_2'$, there must exist an attribute $X \in V'$ that makes e_1 and e_2 give an equal value to the parent of X and e_1 and e_2 give a better value than X. And according to the Ordering Testing (Algorithm 1), we can deduce $N \nvDash e_1 \succ e_2$.

In addition, Lemma 2 shows that an instance e_1 does not dominate(or equal) another instance e_2 on a subset of attributes, then e_1 cannot dominate e_2 on any superset of attributes. So we can use this to get Top-k instances and use depth first search for instances' space.

The effectiveness and the robustness of Algorithm 4 reflected in the following three aspects. Firstly, simulated depth first search is used in the whole process of data retrieval. That is to say, this algorithm can extend the instances which can be satisfied by the database in the top part of the composition. Taking advantage of simulated depth first search, we can quickly find complete instances satisfying the conditions. Secondly, in order to reduce unnecessary instance evaluation, if the partial composition cannot be satisfied by the database, the corresponding extension will not be considered. Finally, as long as an element that belongs to the Top-k subset is identified, it will be output immediately, which greatly reduces the initial response time. However, the Algorithm has some limitations. That is, this method cannot deal with looped CP-nets, because a looped CP-nets will cause a conflict and cannot be used in practice. To guarantee that the generated CP-net is acyclic, we employ an algorithm to generate provably acyclic CP-nets uniformly at random. This algorithm is also computationally efficient and allowed for multi-valued domains [51].

5.1. Indexing mechanism

We mentioned above the top-k depth-first search algorithm, whose efficiency depends on the level of efficiency of satisfiability test. Given an instance of a set of attributes, we need to detect on the database whether exists instances containing the same value on those attribute, which is also a key problem to be solved in the online transaction processing system (OLTP). The two prominent features of OLTP queries are randomness and multi-dimensional, which require equal treatment of each dimension. Therefore, we should allow the users to query multidimensional data in accordance with the combination of any one dimension or several dimensions. We consider following classical indexing methods below.

5.1.1. B+-Tree

B+-Tree [52] is a commonly used tree-structure indexing mechanism in database systems and file systems. It is a variant of B-tree in which all records are stored in the leaves and all leaves are linked sequentially. Non-leaf node is equivalent to a leaf node index and leaf node is equivalent to the data layer for the stored data. B+-tree supports both random search starting from the root node and sequential search. During the search, if a given value is equal to a non-leaf node, we do not terminate, but continue down until reaching a leaf node. Therefore, in the B+-tree, regardless of the search result, each search is taking a path from the root to a leaf node. Internal nodes of the B+-tree do not have the pointer pointing to the keywords for specific information. Thus its internal node relative to the B tree is smaller. If all the same internal node keywords stored in the same disk block, then the disk block can accommodate more number of keywords. There are more keywords by disposable reading into memory, so relatively IO read and write times will be reduced. Furthermore, due to the non-leaf point is not the node pointing to the file, but the index of leaf node keyword. Thus the search path of each keyword must be from the root node to a leaf node. The same keyword query path length will result in same data query efficiency.

The lookup time of each record in B+- tree is basically the same, which needs to go from the root node to the leaf node, and the keyword should be compared again in the leaf node. Because the non-leaf nodes of the B+- tree do not store the actual data, so that the number of elements which each node hold becomes more, but the height of the tree becomes smaller. The advantage of this is to reduce the number of disk access. The time of one disk access of the B+- tree is equivalent to hundreds of times of memory comparison, and the leaf nodes of the B+- tree are connected together by pointers, which contributes to order traversal. For a m-order B+-tree with h levels of index, the first layer has a node, at least two branches, and the second layer has at least 2 nodes. When the number of layers(i) is more than 3, each layer has at least $2 \cdot \lceil \frac{m}{2} \rceil^{i-2}$ nodes. If there are N nodes in the m-order tree, we can deduce that N must satisfy $N \geqslant 2 \cdot \lceil \frac{m}{2} \rceil^{h-1} - 1$. So if the retrieval is successful, the height is $h = 1 + \log\lceil \frac{m}{2} \rceil \cdot \frac{(N+1)}{2}$. h is also the number of disk access. Moreover, the space required to store the tree B+- is $O(n)$.

5.1.2. B+-Tree on a composite key

A B+-Tree on a composite key [53] indexes the key values as if the composite key was a single key, which is similar to B+-Tree. We can see from this structure diagram about B+-Tree above. In Fig. 3, (Bournemouth, 1000) is less than or equal to (Bournemouth, 1000) and so it appears in the first leaf node. However, (Bournemouth, 7500) is greater than (Bournemouth, 1000) and so it appears in the second leaf node. The order of each attribute in the composite key is important, because we need to successively compare each attribute's value to determine the size of keys. For example, although the second value of (Armagh, 1500) is greater than the second value of (Bournemouth, 1000), the order of the attributes means that (Armagh, 1500) is less than (Bournemouth, 1000). Therefore, the above B+-Tree may be used to search for $(branch_{n}ame)or(branch_{n}ame, balance)$ but not (balance). For example, balance = 2000 appears in two paths of the B+-Tree.

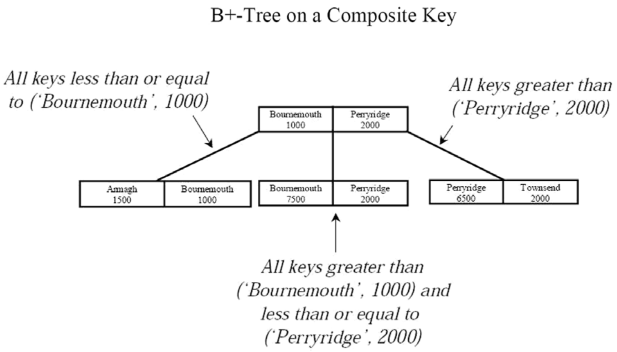

B+-Tree on a Composite Key
All keys less than or equal to ('Bournemouth', 1000)
All keys greater than ('Perryridge', 2000)
Bournemouth 1000
Perryridge 2000
Armagh 1500
Bournemouth 1000
Bournemouth 7500
Perryridge 2000
Perryridge 6500
Townsend 2000
All keys greater than ('Bournemouth', 1000) and less than or equal to ('Perryridge', 2000)

Fig. 3. B+-tree on composite key.

For a B+-Tree on a composite key(B_c) of m-order with n domain values, the number of levels (l) must satisfy $l \leqslant \log_{\lceil \frac{m}{2} \rceil} \frac{N+1}{2} + 1$, and the number of nodes (p) should be $p = \frac{N-1}{\lceil \frac{m}{2} \rceil - l} + 1$. Considering the cost of adding a new relation which contains n_1 attribute values defined on the domain D of the B_c tree and let the existing B_c have n domain values. The total number of attribute values in the resulting composite tree will be $n_c = \mid n \cup n_1 \mid$, and the number of new attribute values to be inserted will be $n_c - n_1$. The average number of times a node will be split per insertion is $\frac{1}{\lceil \frac{m}{2} \rceil - l}$. Let us consider two relations R with n_1 and n_2 separate values for the attribute on which the equijoin is to be performed. Then a composite tree will have at most $n_c = \mid n_1 \cup n_2 \mid$ values and $p_c = 1 + \frac{n-1}{\lceil \frac{m}{2} \rceil - l}$. The total storage space for the internal nodes of the composite tree will be larger than any one of the separate B+- trees. However, due to the absence of duplicates, the space for the internal nodes of the B_c tree will be less than the sum of the space required for the internal nodes of the separate B+- trees. The storage requirements for a B_c tree is between $1 + \frac{n_{\min}-1}{\lceil \frac{m}{2} \rceil - l}$ and $1 + \frac{n_{\max}-1}{\lceil \frac{m}{2} \rceil - l}$ where $n_{\max} = n_1 + n_2$ and $n_{\min} = \max(n_1, n_2)$.

5.1.3. Bitmap

Bitmap [54] is a widely used indexing mechanism for retrieve data sets which meet the attributes of the instance. Bitmap for each possible value of each attribute is appended to an array of bits, where each bit in the array associates an instance in the database. The right part of Fig. 4 shows a conventional bitmap. For an object, if it is given attribute value, the corresponding bit of its bitmap will set to 1, or the bit will be set to 0. To use the conventional bitmap to conduct the satisfiability test, we need to get columns that correspond to the instances, and then do *and* bit operations, which requires a large proportion of the bitmaps.

The size of a Bitmap is $c \times n \times m \times N$ bits. c represents the number of bits for encoding an attribute's value, n is the number of attributes, m is the average number of values for each attribute, and N is the number of instances in the database. The size of Bitmap is as big as that of the database, however, based on algorithm 4, Top-K retrieval usually requires only a small part of Bitmap loading into the memory for satisfiability detection. Therefore, the overall I/O

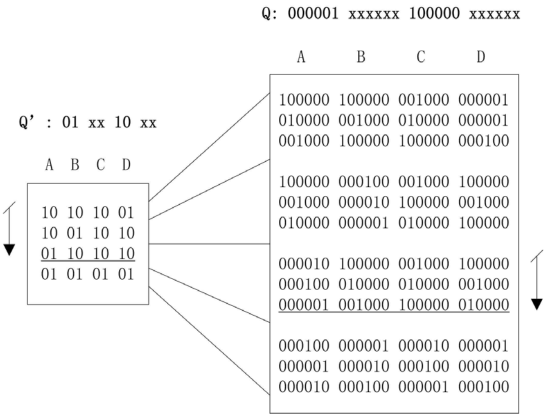


Fig. 4. Bitmap structure.

cost is much smaller than that of sequential scanning. Nevertheless, a drawback of the Bitmap is that the *and* bit operations will be great while meeting large data sets.

5.1.4. Hierarchical bitmap

The hierarchical bitmap [55,56] increases abstraction layers to avoid unnecessary data access on the basis of the original bitmap. The structure of hierarchical bitmap is illustrated on the left of Fig. 4, the abstract level of the hierarchical bitmap is actually a compressed bitmap, which combines the columns of each attribute into a smaller number of columns. To perform the merge, we use a bitwise OR on the columns being merged. (While in Fig. 4, it illustrates only one level of abstraction, multiple abstract levels can be used in practice, where each level is an abstraction of another). To test the satisfiability of an instantiation, the hierarchical bitmap is used. Then the system starts with a scan of the most abstract level. When there is a row R that satisfies the requirement, it will descend to the more concrete levels to evaluate the rows that correspond to R. The process will stop when the system descends to the original bitmap and obtain a row that satisfies the requirement or scanning the entire hierarchical bitmap. To better clarify it, we put forward an example. To test the satisfiability of $(A = a_6) \wedge (C = c_1)$, firstly, we transfer the instantiation to the corresponding bit strings. As illustrated in Fig. 4, Q and Q′ are the corresponding bit strings for the original abstract bitmaps. Then the system will scan the abstract bitmap, comparing the Q′ with each row. When a match is identified, then the system will scan the corresponding proportion of the original bitmap and compare this rows with Q. If in this proportion, a row is same with the Q, we can obtain that $(A = a_6) \wedge (C = c_1)$ is satisfied and the process will be stopped. When the whole bitmaps are scanned and no row is matched with Q′, we can get that $(A = a_6) \wedge (C = c_1)$ is not satisfied. As illustrated in Fig. 4, we can finish the satisfiability test of $(A = a_6) \wedge (C = c_1)$ by evaluating 6 rows, 3 in the abstract bitmap and 3 in the original bitmap.

Intuitively, regardless of whether the test instances occur frequently in database, it is better to use the hierarchical Bitmap to do satisfiability tests. If the instances occur frequently in data sets, the scanning process is likely to stop at an early stage. If the instances seldom occur in data sets, then the algorithm can skip the data which is before matching rows in original Bitmap.

To retrieve the top-k results, a number of satisfiability tests are conducted in Algorithm 4. Normally the evaluated data instantiations share a significant number of attribute values because the algorithm conducts the tests in the depth-first order. Therefore, we can allow each satisfiability test to reuse the bitmap access of previous tests to further improve the efficiency of top-k retrieval. In Algorithm 4, when one instantiation passes the satisfiability test, it will be pushed back to the stack (i.e. ST) and extended with an additional attribute. Then the system will mark the position that the test has already reached in the bitmap, and the new instantiation in the stack with the marks will be stored. When a new instance is tested, we can scan the bitmap from the mark position.

H-Bitmap can avoid a large number of unnecessary visits to the original Bitmap. Assuming that there are D attributes, each attribute can take C possible values. In other words, D and C respectively represent the dimensions and cardinality of data queries. So, each of the lines at the bottom of H-Bitmap contains $D \times C$ bit. We merge every m column attributes into one column to construct the abstract Bitmap. The abstraction process continues until two columns left. In this case, we finally got the log_m^c layers. The instance to test is set to be 1, which contains t properties. Firstly, we assume that there is no abstract layer, then the algorithm must scan the original Bitmap until the first row matching instance 1 is found. In this case, the expected value of the number of bit that needs to be accessed is $C^t \times D \times C$. After using the abstract layer, some of the rows in the original Bitmap will be skipped, and the expected value of the number of bit that needs to be accessed in original Bitmap is $m^t \times D \times C$. Then we consider the expected value to be accessed in the first abstract layer in H-Bitmap. If we assume that there are no other abstract layers above this level, this expected value can be calculated to be $(\frac{C}{D})^{t+1} \times D$. Applying this calculation process to all abstract levels, we can get the expected number of bit numbers to be accessed in the satisfiability test is $m^t \times D \times C \times \sum_{i=0}^{log_m^c} m^{-t}$. Since $\sum_{i=0}^{log_m^c} m^{-1}$ is between 1 to 2, the expected number mainly is $m^t \times D \times C$. As we can see, unless the dimension is too high, the overhead of algorithm of H-Bitmap can be relatively small.

6. Experiments and analysis

In this section, we have conducted an extensive set of experiments to evaluate the feasibility and effectiveness of the proposed approaches. Each group of experiments was performed for 10 times. We report the mean value (μ) of the ten runs and their standard deviation (σ). In addition, the statistical significance test (p) is performed for comparison experiments.

The flow chart below shows the main process. According to Fig. 5, first of all, we input the dimension of CP-net. Then we judge the correlation of the attributes. And when the correlation between the data dimension is determined, we firstly exploit Hilbert Curve to make high multidimensional data into one-dimensional, and then compare the efficiency of Hilbert Curve plus bitmap and Hilbert Curve plus B+-tree. When the correlation between the data dimensions is non-determined, we utilize PCA to make high-dimensional data into low dimensional data in the first place. We then compare the performance of PCA plus bitmap and PCA plus B+-tree.

In our experiments, we used both synthetic and real data. The real data set is the QWS data set.[1] In this data set, a Web Service Crawler is used to collect QoS attributes of real web services. It contains 2507 Web Services' QoS attributes including, Response Time, Availability, Throughput, Successability, Reliability, Compliance, Best Practices, Latency, Documentation, Service Name, and WSDL Address. Besides the real data, we also synthetically generate testing data, which allows us to change the characteristics of a data set and observe the performance of the top-k approaches in different circumstances. Based on the combination of both QWS data sets and synthetic data sets, the experiments can be carried out.

[1] http://www.uoguelph.ca/~7eqmahmoud/qws/.

In both synthetic and real data set, user preferences are represented by a series of randomly generated CP-nets. Each generated CP-net is a random partial order graph, among them the number of the parent nodes varied from 0 to 3. Also, to make the order of attribute values randomly distribute, the CPTs of CP-nets are also randomly generated.

6.1. Correlation between the data dimensions is determined

In the case of correlation between the data dimensions being determined, we compared three indexing methods, two of which employ Hilbert dimension reduction. They are dimensionality reduction plus B+tree, dimensionality reduction plus Bitmap, and Composite Key B+tree. Figs. 6, 7, 8, 9, 10, 11 summarize the experimental results. And during these experiments, the size of the data set defaults to 10 000.

Overall, when the cardinality and k are determined, the higher dimension the longer the response time. As can be seen from Figs. 6, 7, and 8, when k is five, different dimensions perform similarly and the response time increases with the increasing k. When the dimension is determined, the cardinality affects the performance of bitmap indexes whose efficiency decreases rapidly when the cardinality is larger than 20. When cardinality is 10, the three methods have little difference. When the cardinality reaches 20, the curves tend to be different. In sum, composite key outperforms the other two methods.

Similarly, when the dimension and k are determined, the higher the cardinality is, the longer response time. When the dimension and k are determined as shown in Figs. 9, 10, 11, cardinality changes have little effect on the response time. However, different values of k still play an important role in the experimental result. For a cardinality lower than 10, the performance of composite key is ordinary, but improves with the increasing cardinality.

Note that the efficiency of composite key exceeds the two reduction dimensional indices mainly because the time cost of Hilbert curve dimension reduction encoding and decoding surpasses the benefits of dimension reduction. Along with the increase of dimension, the retrieval efficiency of the two dimension reduction methods starts to outperform the composite key.

6.2. Non-determined data correlation

Under the condition of correlation between the data dimensions being non-determined, we compare the indexes using PCA dimension reduction with the original bitmap method. Figs. 12, 13, 14, 15, 16, 17, summarize the experimental results. And during these experiments, the size of the data set defaults to 10 000.

The PCA plus Bitmap method is overall superior to the original multidimensional Bitmap. In the experiment, we set dimensions as 3-D, 6-D, and 9-D, and then compare the time based on cardinality of 10, 20, 50. We can see from the result, in the low-dimensional case (3-D), the efficiency is similar. However, in the 6-D and 9-D, PCA plus bitmap outperforms the other two methods significantly for cardinality of ten and twenty. Although the PCA plus Bitmap is still better than the original Bitmap in the cardinality of fifty, the advantage of the PCA plus Bitmap is not evident as other cases. This can be explained as follows. The original Bitmap performance has been adequate in low dimensional (3-D) itself. So it makes no distinction. However, in high dimensional cases, PCA dimensionality reduction can speed up retrieval effectively. Furthermore, cardinality plays an important role in response time. As we can see, when the cardinality is less than 20, the performance of Bitmap is good. But it decreases rapidly when the cardinality exceeds 20.

As for PCA plus Hierarchical Bitmap, intuitively, it should be more efficient. However, the result is not always the case. We can see from the figure, when we set the dimension as 3-D, PCA plus Hierarchical Bitmap even gives the worst performance. In 6-D, it is slightly better than the original bitmap and worse than PCA plus bitmap. This is because that Hierarchical Bitmap increases the amount of i/o times which drags down the overall performance.

Here, we also expand the data set and conduct some comparative experiments. And during these experiments, we adjust the size of the data set from 2500 to 50 000. The experimental results are illustrated in Fig. 18, and from the figure, we can see that the dimensionality reduction has a good performance on reducing the response time.

6.3. Top-k methods comparison

In this section, we conduct a series of experiments to compare the performance of different top-k methods. The first method uses sequential scan without any indexing mechanism. This method is an intuitive algorithm for Top-k retrieval. It scans the data items in the database and outputs the Top-k data sets that are sorted according to the ordering test. However, this algorithm costs too much for large data sets. The reason is that even if you only retrieve one best data item, you need to scan the entire database. The second method uses the utility function based top-k method to scan the database. The utility function adopts the idea of quick sort, which divides the data recursively without searching the entire database. Using the utility function to select the *k* elements with the highest score in the data set, the most direct method is to construct a small heap of *k* size and scan the elements of the data set one by one. If the current element's score is higher than that of the top of the pile, it will be replaced. Then adjust the top stack to maintain the property of "small top stack". After scanning the data set, the elements in the small top stack are the *k* elements with the highest scores. We use the idea of quick sort, which can be divided by recursion. In this way, the initial response time of the algorithm will be improved in some cases. Because it does not need to wait until the complete data set is scanned. The third method uses the depth first retrieval introduced in our paper. To avoid redundant experiment, we only use the PCA plus bitmap to reduce the dimension.

In the first set of experiments, we fix the number of services as 7.5k, and the cardinality of attributes as 8. Fig. 19 shows the result of efficiency comparison of different top-k methods on different dimension. It can be seen that the response time of sequential scan does not vary much with the increase of dimensionality. This is because sequential search just needs to scan the total database without considering the dimensionality of attributes. In contrast, for utility function method and depth first retrieval, increasing the dimensionality will increase the sparsity of the data, which forces the algorithm to perform more satisfiability tests. The increase of dimensionality also directly increases the depth of the search space for the depth first algorithm.

In the second set of experiments, we fix the number of services as 7.5k, and the dimensionality of data as 5. The result of the efficiency comparison of different methods on the different number of cardinality is shown in Fig. 20. It can be seen that the cost of all three methods increase as the increase of the number of cardinality of attributes. However, the cost of sequential scan grows faster than other two methods. This is because sequential scan needs to search the entire database, and cardinality increase leads to the increase of records. For the other two methods, they need not to search the entire database and hence performs more efficiently than sequential scan.

In the third set of experiments, we fix the dimensionality of data as 5, and the cardinality of attributes as 8. Additionally, we adjust the size of the data set from 2500 to 15 000. The result of the efficiency comparison of number of services on different top-k methods is illustrated in Fig. 21. It can be seen that the cost of all

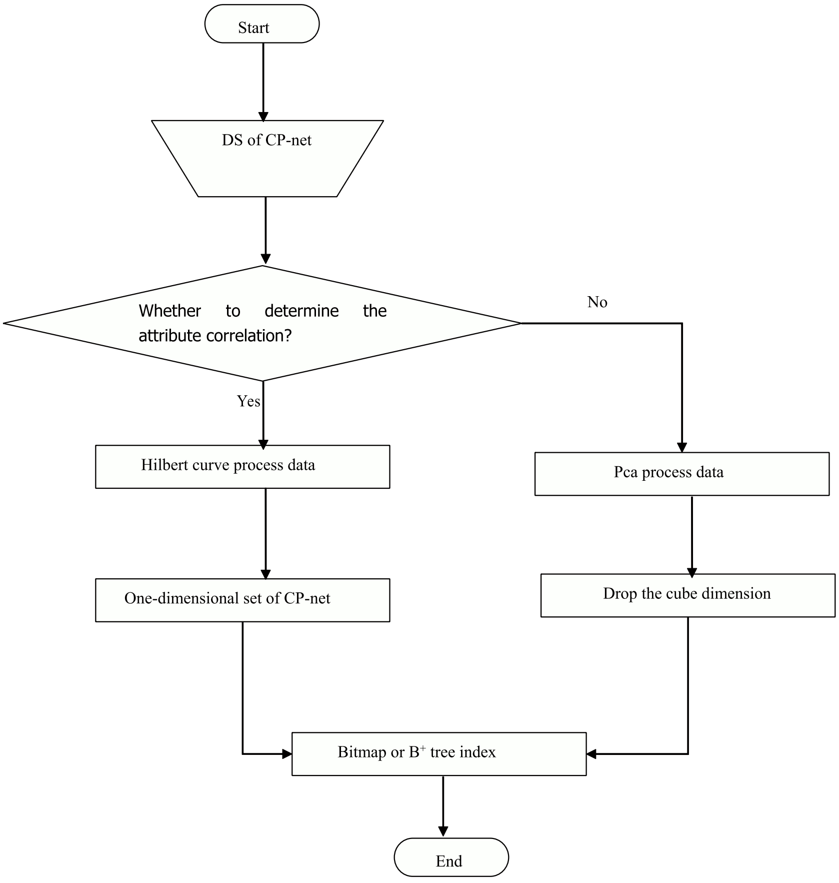
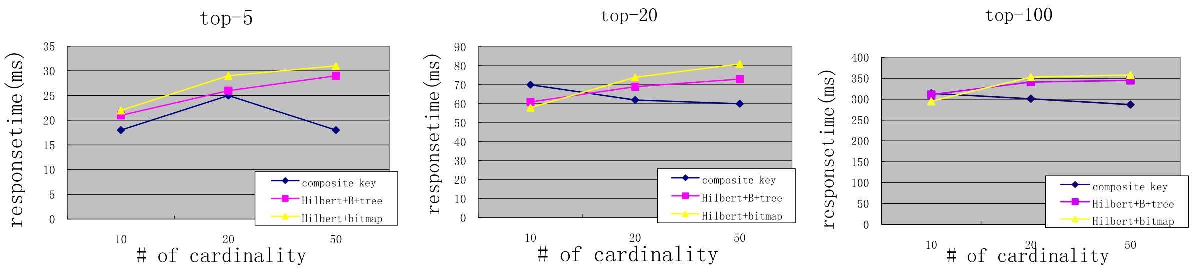
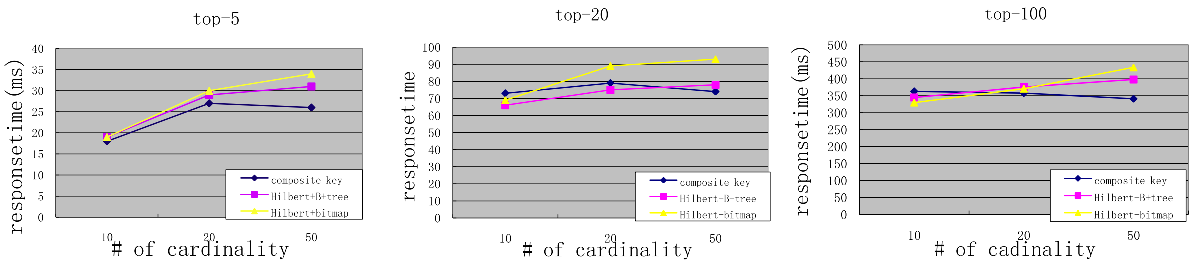

Start
DS of CP-net
Whether to determine the attribute correlation?
No
Yes
Hilbert curve process data
Pca process data
One-dimensional set of CP-net
Drop the cube dimension
Bitmap or B+ tree index
End

Fig. 5. Main flow chart.



(a) Top-5 (b) Top-20 (c) Top-100

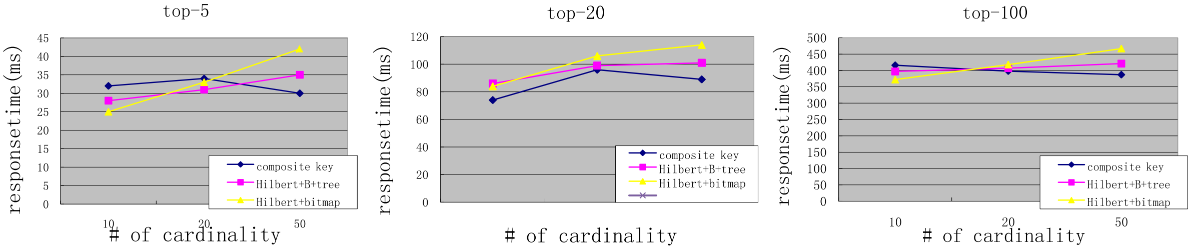
Fig. 6. The efficiency of indexs for Top-k on three dimension.



(a) Top-5 (b) Top-20 (c) Top-100

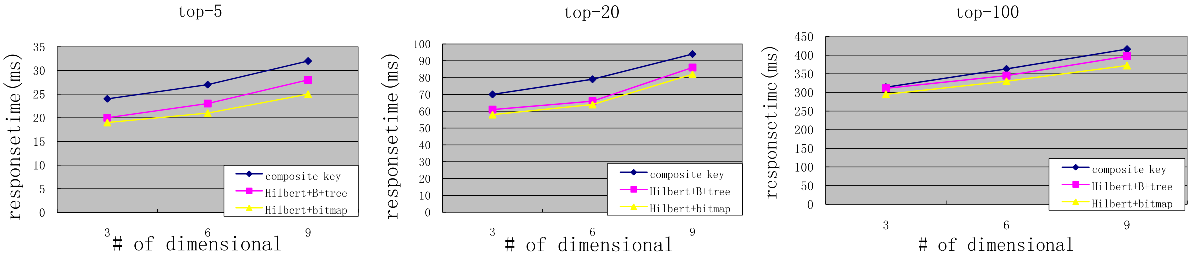
Fig. 7. The efficiency of indexs for Top-k on six dimension.

three methods will increase with the increase of size of services. But the value of k will not influence the performance of sequential scan. As sequential scan needs to go through the complete data set before generating the top-k result, its response time increases



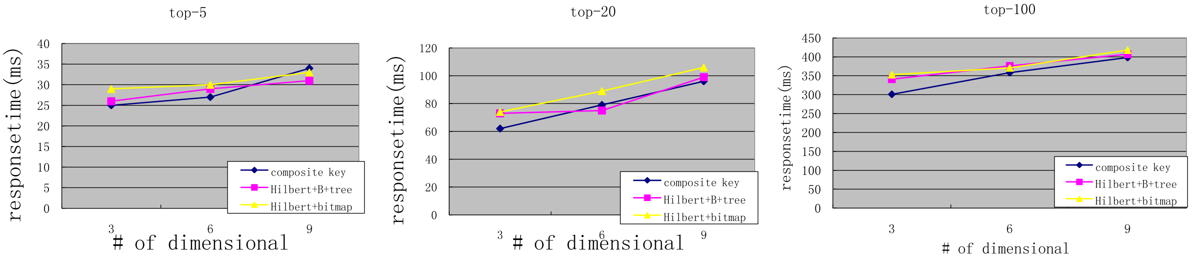
(a) Top-5 (b) Top-20 (c) Top-100

Fig. 8. The efficiency of indexs for Top-k on nine dimension.



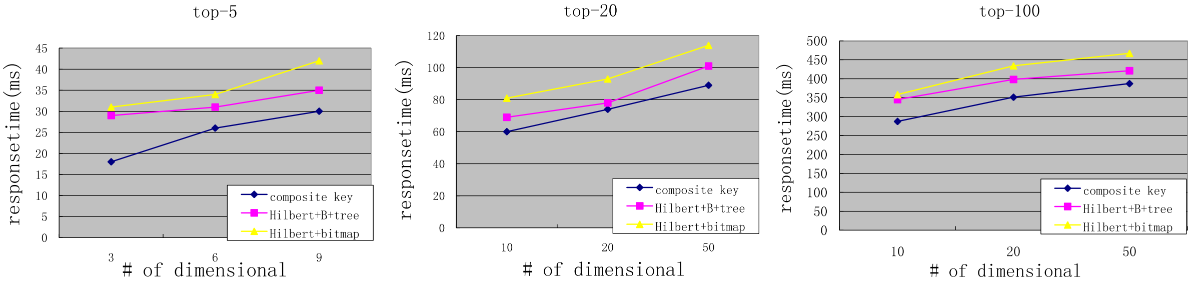
(a) Top-5 (b) Top-20 (c) Top-100

Fig. 9. The efficiency of indexs for Top-k on ten cardinality.



(a) Top-5 (b) Top-20 (c) Top-100

Fig. 10. The efficiency of indexs for Top-k on twenty cardinality.



(a) Top-5 (b) Top-20 (c) Top-100

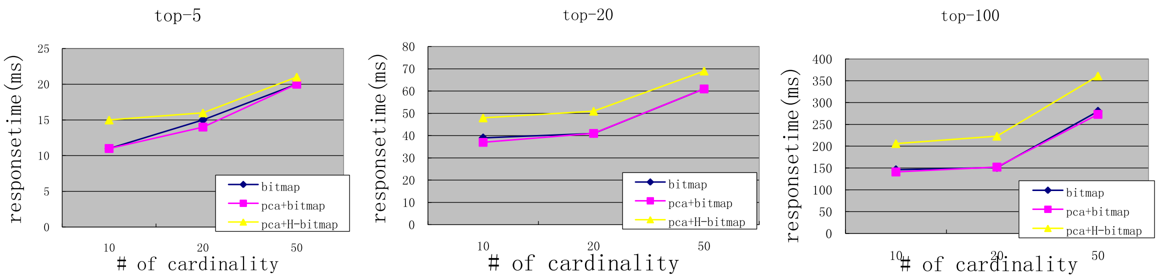
Fig. 11. The efficiency of indexs for Top-k on fifty cardinality.

linearly with the size of the data set. Its performance remains the same no matter what k is.

6.4. *Dimensionality reduction methods comparison*

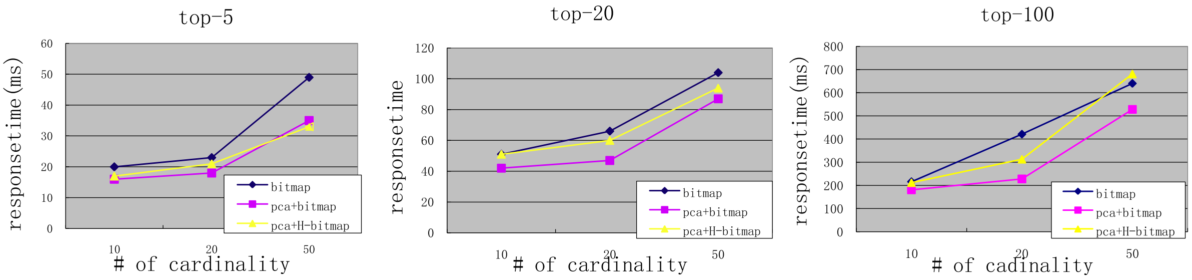
In this section, we conduct a series of experiments to compare the performance of different dimensionality reduction methods. The first method uses original bitmap without any dimensionality reduction methods. The second method uses the Hilbert curve dimension reduction with original bitmap. The third method uses the PCA dimension reduction with original bitmap. To avoid redundant experiments, we only use the original bitmap index mechanism.

In the first set of experiments, we fix the dimensionality of data as 6 and the cardinality of attributes as 20. In addition, we adjust the size of the data set from 2500 to 50 000. Moreover, these experiments are done under the condition that the correlation between the data dimensions is determined. In Figs. 24–26, we report the response time in terms of the mean value (μ) of the ten runs, their standard deviation (σ), and the significant difference (p) for different methods. We can see that p is almost to 1, which



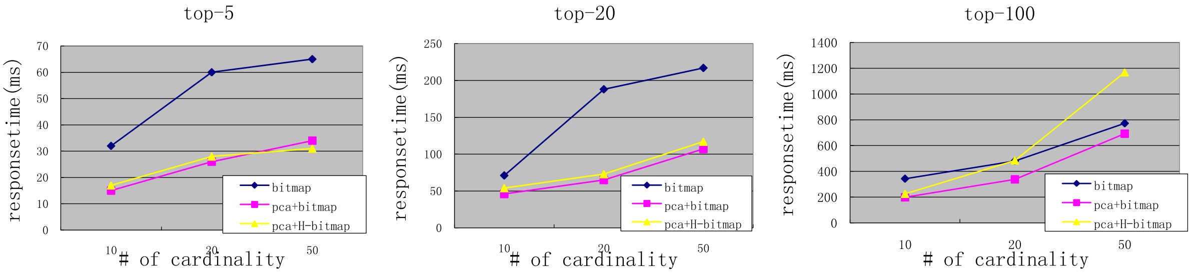
(a) Top-5 (b) Top-20 (c) Top-100

Fig. 12. The efficiency of indexs for Top-k on three dimension.



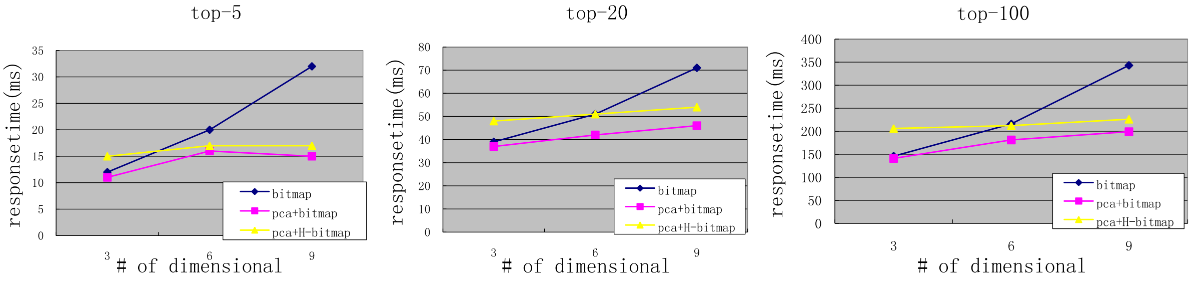
(a) Top-5 (b) Top-20 (c) Top-100

Fig. 13. The efficiency of indexs for Top-k on six dimension.



(a) Top-5 (b) Top-20 (c) Top-100

Fig. 14. The efficiency of indexs for Top-k on nine dimension.

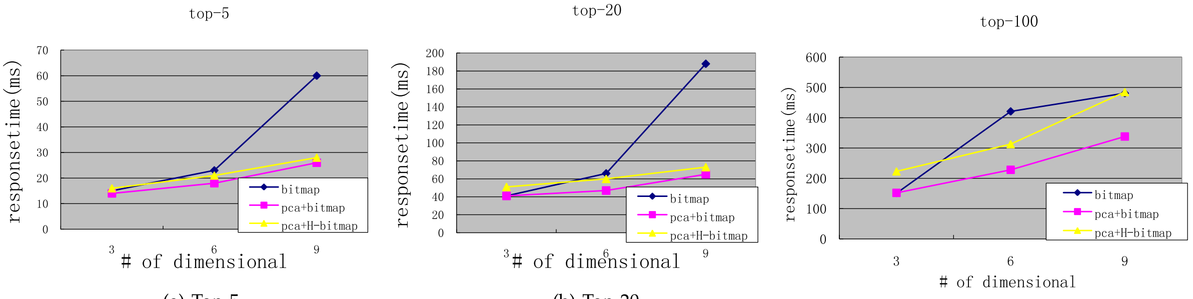


(a) Top-5 (b) Top-20 (c) Top-100

Fig. 15. The efficiency of indexs for Top-k on ten cardinality.

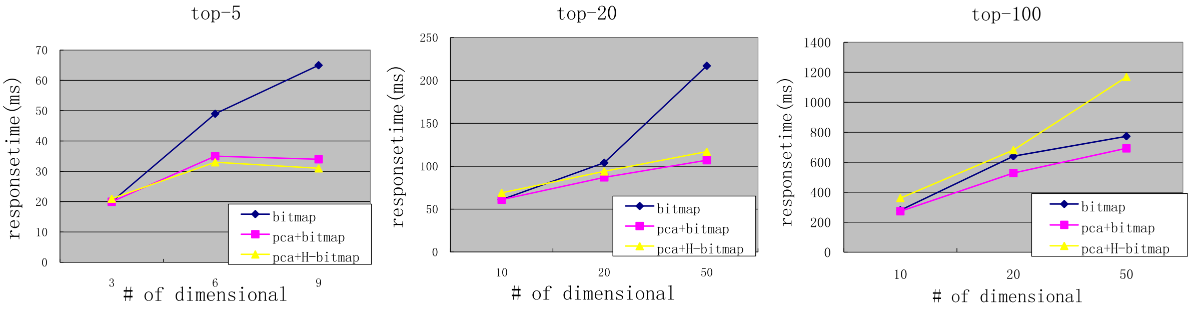
indicates the data of ten runs is of slight difference. Therefore, we use the μ for drawing the figures. And Fig. 22 shows the result of efficiency comparison of different dimensionality reduction methods on the size of the data set. It can be seen that the response time of bitmap method is the most among the three methods. Hilbert curve plus bitmap method and PCA plus bitmap method are both overall superior to the original multidimensional bitmap. This is because that original bitmap ignores dimensionality reduction which helps to reduce the times of retrieval process and improve retrieval efficiency. What is more, we can see that Hilbert curve plus bitmap method works better than PCA plus bitmap method, which reflects that Hilbert curve is more suitable than PCA when the correction is determined. The reason may be that Hilbert curve processing data set does not affect the information of the original data set.

In the second set of experiments, we fix the dimensionality of data as 6 and the cardinality of attributes as 20. In addition,



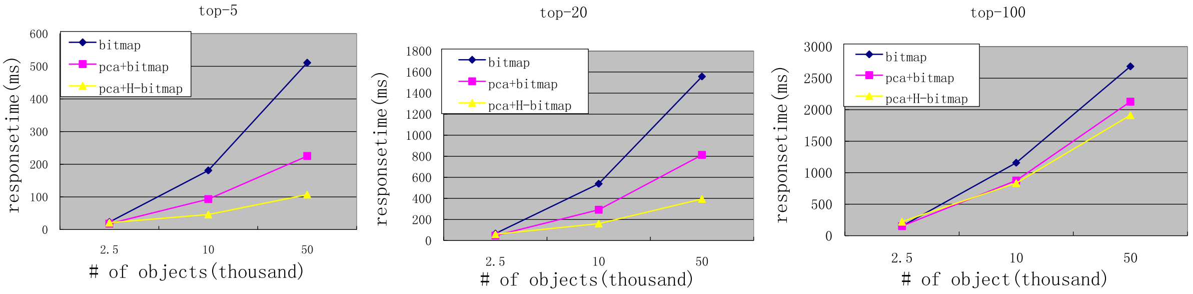
(a) Top-5 (b) Top-20 (c) Top-100

Fig. 16. The efficiency of indexs for Top-k on twenty cardinality.



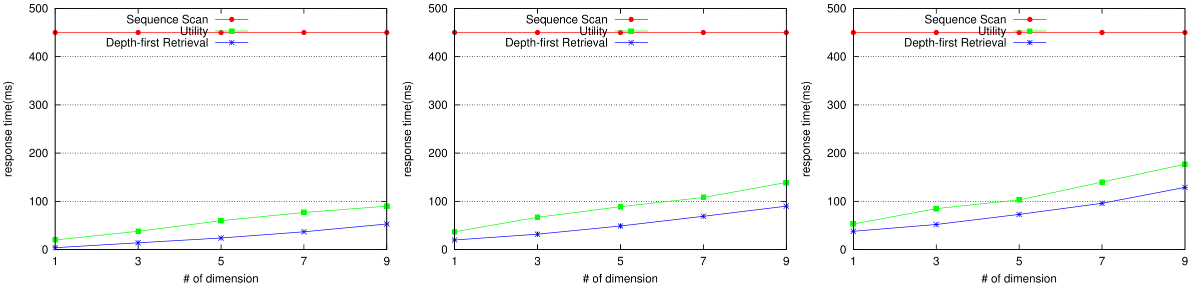
(a) Top-5 (b) Top-20 (c) Top-100

Fig. 17. The efficiency of indexs for Top-k on fifty cardinality.



(a) Top-5 (b) Top-20 (c) Top-100

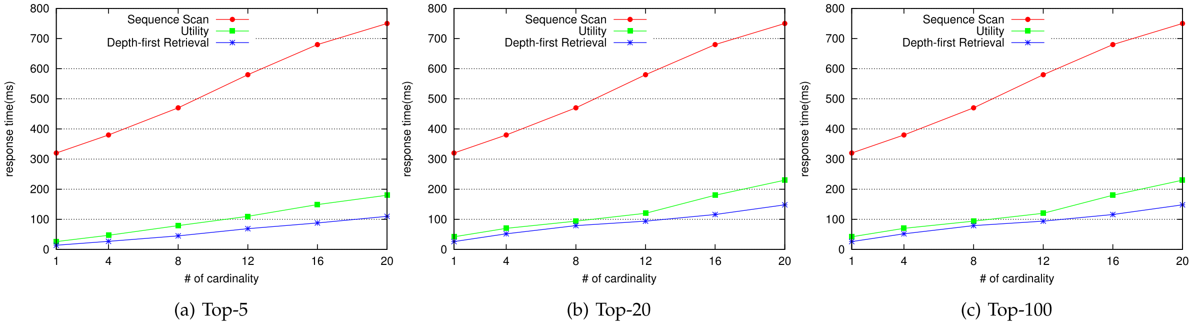
Fig. 18. The efficiency of indexs for different data sets on six dimension, twenty cardinality.



(a) Top-5 (b) Top-20 (c) Top-100

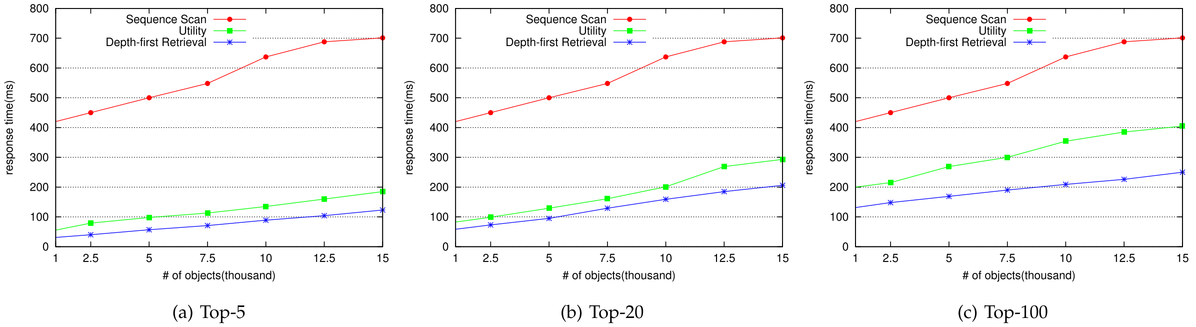
Fig. 19. The efficiency comparison of different top-k methods on the dimension.

we adjust the size of the data set from 2500 to 50 000. Moreover, these experiments are done under the condition that the correlation between the data dimensions is uncertain. The result of the comparison of different dimensionality reduction methods on the size of the data set is shown in Fig. 23. Similar to the first set of experiments, it is obvious that bitmap method still works the worst. However, the response time of PCA plus bitmap method is less than Hilbert curve plus bitmap method. This could be explained that PCA can be used to determine the correlation between data attributes and optimize data processing. Therefore, it is better to use PCA than Hilbert curve when the correction is uncertain.



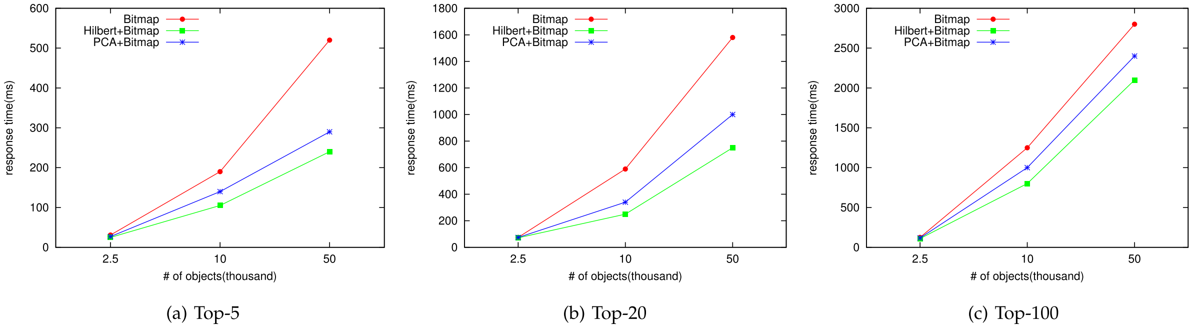
(a) Top-5 (b) Top-20 (c) Top-100

Fig. 20. The efficiency comparison of different top-k methods on the cardinality.



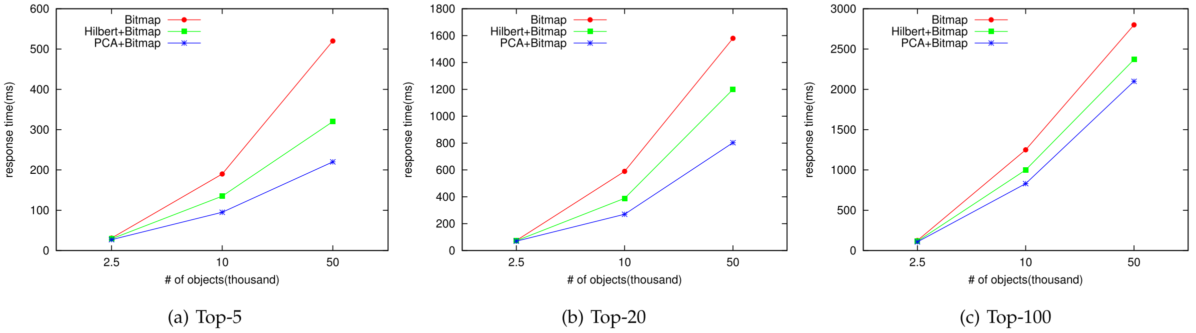
(a) Top-5 (b) Top-20 (c) Top-100

Fig. 21. The efficiency comparison of different top-k methods on the size of services.



(a) Top-5 (b) Top-20 (c) Top-100

Fig. 22. The efficiency comparison for different dimensionality reduction methods when the correlation is determined.



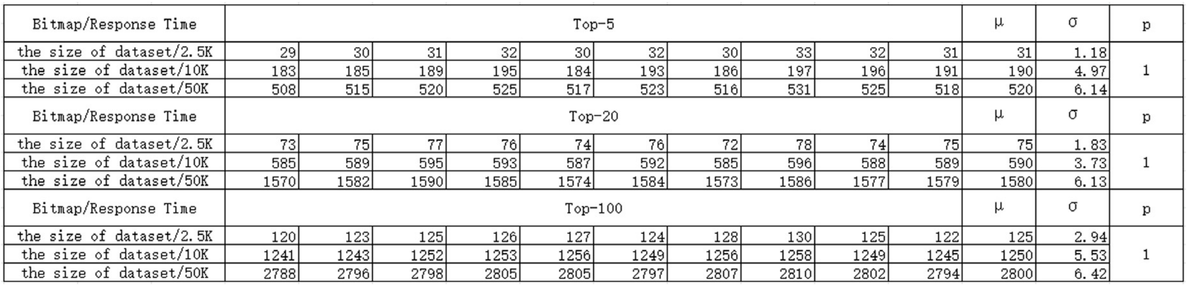
(a) Top-5 (b) Top-20 (c) Top-100

Fig. 23. The efficiency comparison for different dimensionality reduction methods when the correlation is uncertain.

7. Conclusion and future work

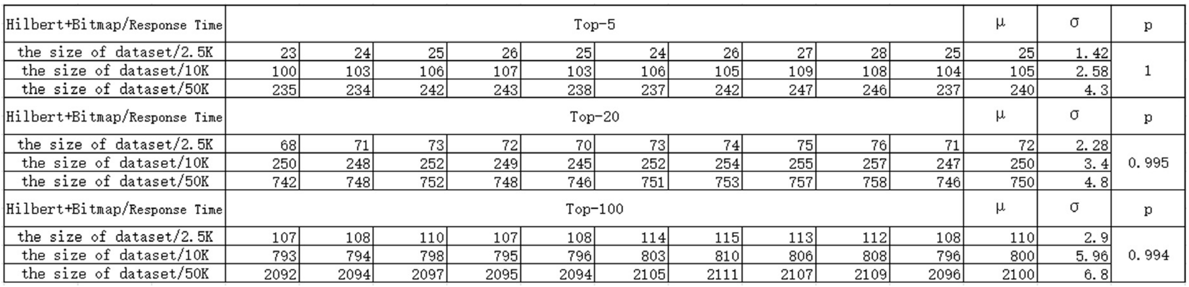
7.1. Conclusion

In this paper, we develop effective indexing methods for top-k retrieval using Conditional Preference Network (CP-Net). We divide indexing methods of Top-k retrieval for CP-Net into two categories based on whether the correlation between the data dimension is determined. In order to increase the retrieval efficiency, we introduce two dimension reduction methods, Hilbert Curve and PCA. When the correlation between the data dimension is determined, we firstly exploit Hilbert Curve to make multi-dimensional data into one-dimensional data, and then compare the efficiency of composite key B+-tree, Hilbert Curve plus bitmap and Hilbert



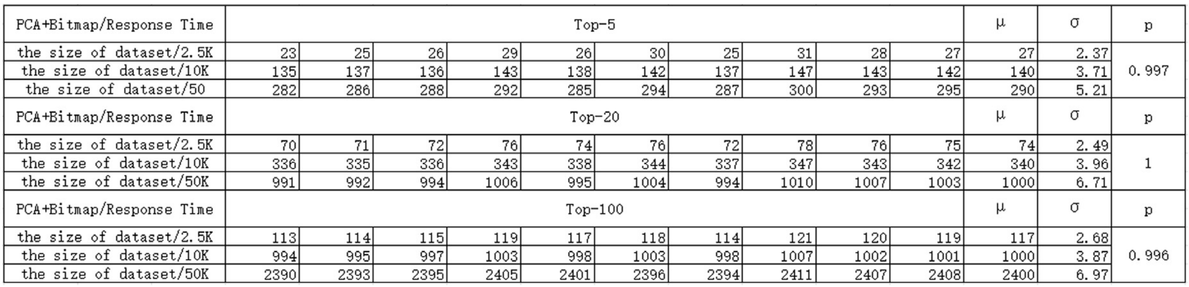
Bitmap/Response Time	Top-5										μ	σ	p
the size of dataset/2.5K	29	30	31	32	30	32	30	33	32	31	31	1.18	1
the size of dataset/10K	183	185	189	195	184	193	186	197	196	191	190	4.97	
the size of dataset/50K	508	515	520	525	517	523	516	531	525	518	520	6.14	
Bitmap/Response Time	Top-20										μ	σ	p
the size of dataset/2.5K	73	75	77	76	74	76	72	78	74	75	75	1.83	1
the size of dataset/10K	585	589	595	593	587	592	585	596	588	589	590	3.73	
the size of dataset/50K	1570	1582	1590	1585	1574	1584	1573	1586	1577	1579	1580	6.13	
Bitmap/Response Time	Top-100										μ	σ	p
the size of dataset/2.5K	120	123	125	126	127	124	128	130	125	122	125	2.94	1
the size of dataset/10K	1241	1243	1252	1253	1256	1249	1256	1258	1249	1245	1250	5.53	
the size of dataset/50K	2788	2796	2798	2805	2805	2797	2807	2810	2802	2794	2800	6.42	

Fig. 24. Mean value, standard deviation and significant difference/first method in Dimensionality Reduction methods comparison.



Hilbert+Bitmap/Response Time	Top-5										μ	σ	p
the size of dataset/2.5K	23	24	25	26	25	24	26	27	28	25	25	1.42	1
the size of dataset/10K	100	103	106	107	103	106	105	109	108	104	105	2.58	
the size of dataset/50K	235	234	242	243	238	237	242	247	246	237	240	4.3	
Hilbert+Bitmap/Response Time	Top-20										μ	σ	p
the size of dataset/2.5K	68	71	73	72	70	73	74	75	76	71	72	2.28	0.995
the size of dataset/10K	250	248	252	249	245	252	254	255	257	247	250	3.4	
the size of dataset/50K	742	748	752	748	746	751	753	757	758	746	750	4.8	
Hilbert+Bitmap/Response Time	Top-100										μ	σ	p
the size of dataset/2.5K	107	108	110	107	108	114	115	113	112	108	110	2.9	0.994
the size of dataset/10K	793	794	798	795	796	803	810	806	808	796	800	5.96	
the size of dataset/50K	2092	2094	2097	2095	2094	2105	2111	2107	2109	2096	2100	6.8	

Fig. 25. Mean value, standard deviation and significant difference/second method in Dimensionality Reduction methods comparison.



PCA+Bitmap/Response Time	Top-5										μ	σ	p
the size of dataset/2.5K	23	25	26	29	26	30	25	31	28	27	27	2.37	0.997
the size of dataset/10K	135	137	136	143	138	142	137	147	143	142	140	3.71	
the size of dataset/50	282	286	288	292	285	294	287	300	293	295	290	5.21	
PCA+Bitmap/Response Time	Top-20										μ	σ	p
the size of dataset/2.5K	70	71	72	76	74	76	72	78	76	75	74	2.49	1
the size of dataset/10K	336	335	336	343	338	344	337	347	343	342	340	3.96	
the size of dataset/50K	991	992	994	1006	995	1004	994	1010	1007	1003	1000	6.71	
PCA+Bitmap/Response Time	Top-100										μ	σ	p
the size of dataset/2.5K	113	114	115	119	117	118	114	121	120	119	117	2.68	0.996
the size of dataset/10K	994	995	997	1003	998	1003	998	1007	1002	1001	1000	3.87	
the size of dataset/50K	2390	2393	2395	2405	2401	2396	2394	2411	2407	2408	2400	6.97	

Fig. 26. Mean value, standard deviation and significant difference/third method in Dimensionality Reduction methods comparison.

Curve B+-tree. When the correlation between the data dimensions is non-determined, we utilize PCA to make high-dimensional data into low-dimensional data in the first place. We then compare the performance of Bitmap, PCA plus bitmap and PCA plus Hierarchical Bitmap. We conduct an extensive experimental study on both real and synthetic data to demonstrate the effectiveness of those indexing methods, especially on large-scale high-dimensional data sets.

7.2. Future work

This paper investigates the effectiveness and feasibility of traditional and reduction-dimensional indexing mechanisms on large high-dimensional data sets. Traditional indexing research has been quite mature, but the indexing methods on satisfiability testing, particularly big high-dimensional data sets exhibit high computational cost. Therefore, improving the traditional indexing approaches becomes important. This paper reduces the dimensionality of the data sets and then builds the indexes. The choice for dimensionality reduction methods have a direct impact on data retrieval. We summarize the future work as follows:

- The dimensionality reduction methods used in this paper have their limitations. For example, PCA extracts global features of the data sets, and will cause some loss of information. Wavelet transform is mainly used to extract local features but ignores the overall characteristics. So, further work may consider to use other dimension reduction methods and compare their effectiveness.
- The selected dimensionality reduction methods help improve the efficiency of search results. Meanwhile, they may also affect recommendation correctness. Different dimensionality reduction methods may have different impact on the search results and the accuracy of the recommendations. Therefore, assessing the error rate for each dimensionality reduction method is an interesting direction, which will be considered in our future work.

Acknowledgments

This work was partially supported by National Key Research and Development Plan, China (No. 2018YFB1003800) and NSFC Projects, China (Nos. 61672152, 61532013), Collaborative Innovation Centers of Novel Software Technology and Industrialization and Wireless Communications Technology, China. Qi Yu's work was supported in part by an NSF IIS, United States award IIS-1814450 and an ONR, United States award N00014-18-1-2875. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agency.

References

[1] R. Fagin, A. Lotem, M. Naor, Optimal aggregation algorithms for middleware, in: Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, ACM, 2001, pp. 102–113.

[2] S. Chaudhuri, L. Gravano, Evaluating top-k selection queries, in: Proceedings of the International Conference on Very Large Data Bases, 1999, pp. 399–410.

[3] I.F. Ilyas, W.G. Aref, A.K. Elmagarmid, Supporting top-k join queries in relational databases, VLDB J. Int. J. Very Large Data Bases 13 (3) (2004) 207–221.

[4] I.F. Ilyas, G. Beskales, M.A. Soliman, A survey of top-k query processing techniques in relational database systems, ACM Comput. Surv. 40 (4) (2008) 11.

[5] S. Borzsony, D. Kossmann, K. Stocker, The skyline operator, in: Data Engineering, 2001. Proceedings. 17th International Conference on, IEEE, 2001, pp. 421–430.

[6] K.-L. Tan, P.-K. Eng, B.C. Ooi, et al., Efficient progressive skyline computation, in: Proceedings of the International Conference on Very Large Data Bases, 2001, pp. 301–310.

[7] D. Kossmann, F. Ramsak, S. Rost, Shooting stars in the sky: an online algorithm for skyline queries, in: Proceedings of the 28th International Conference on Very Large Data Bases, VLDB Endowment, 2002, pp. 275–286.

[8] Y. Yuan, X. Lin, Q. Liu, W. Wang, J.X. Yu, Q. Zhang, Efficient computation of the skyline cube, in: Proceedings of the 31st International Conference on Very Large Data Bases, VLDB Endowment, 2005, pp. 241–252.

[9] R.C.-W. Wong, A.W.-C. Fu, J. Pei, Y.S. Ho, T. Wong, Y. Liu, Efficient skyline querying with variable user preferences on nominal attributes, Proc. VLDB Endowment 1 (1) (2008) 1032–1043.

[10] C. Boutilier, R.I. Brafman, H.H. Hoos, D. Poole, Reasoning with conditional ceteris paribus preference statements, in: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc., 1999, pp. 71–80.

[11] C. Boutilier, R.I. Brafman, C. Domshlak, H.H. Hoos, D. Poole, Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements, J. Artif. Intell. Res. 21 (2004) 135–191.

[12] J. Liu, S. Liao, Expressive efficiency of two kinds of specific cp-nets, Inform. Sci. 295 (2015) 379–394, [Online]. Available: http://dx.doi.org/10.1016/j.ins.2014.10.038.

[13] S. Coste-Marquis, J. Lang, P. Liberatore, P. Marquis, Expressive power and succinctness of propositional languages for preference representation, in: Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004), Whistler, Canada, June 2-5, 2004, 2004, pp. 203–212, [Online]. Available: http://www.aaai.org/Library/KR/2004/kr04-023.php.

[14] J. Goldsmith, J. Lang, M. Truszczynski, N. Wilson, The computational complexity of dominance and consistency in cp-nets, 2005.

[15] C. Domshlak, F. Rossi, K.B. Venable, T. Walsh, Reasoning about soft constraints and conditional preferences: complexity results and approximation techniques, arXiv preprint arXiv:0905.3766, 2009.

[16] H. Wang, X. Zhou, W. Chen, P. Ma, Top-k retrieval using conditional preference networks, in: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, ACM, 2012, pp. 2075–2079.

[17] D. Hilbert, Ueber die stetige abbildung einer line auf ein flächenstück, Math. Ann. 38 (3) (1891) 459–460.

[18] H. Hotelling, Analysis of a complex of statistical variables into principal components, J. Educ. Psychol. (1933) 498–520.

[19] F. Wenzel, W. Kießling, A preference-driven database approach to reciprocal user recommendations in online social networks, in: Database and Expert Systems Applications:27th International Conference, DEXA Springer International Publishing, Cham, 2016, pp. 3–10.

[20] M. Sarwat, R. Moraffah, M.F. Mokbel, J.L. Avery, Database system support for personalized recommendation applications, in: 2017 IEEE 33rd International Conference on Data Engineering (ICDE), 2017, pp. 1320–1331.

[21] J. Chomicki, Preference formulas in relational queries, ACM Trans. Database Syst. 28 (4) (2003) 427–466.

[22] J. Li, Y. Yan, D. Lemire, Scaling up web service composition with the skyline operator, in: 2016 IEEE International Conference on Web Services (ICWS), 2016, pp. 147–154.

[23] Y.L. Hsueh, C.C. Lin, C.C. Chang, An efficient indexing method for skyline computations with partially ordered domains, IEEE Trans. Knowl. Data Eng. 29 (5) (2017) 963–976.

[24] K. Benouaret, I. Benouaret, M. Barhamgi, D. Benslimane, Top-k cloud service plans using trust and qos, in: 2017 IEEE International Conference on Services Computing (SCC), 2017, pp. 507–510.

[25] L. Purohit, S. Kumar, A classification based web service selection approach, IEEE Trans. Serv. Comput. PP (99) (2018) 1–1.

[26] C. Wang, J. Arenson, F. Helff, L. Gruenwald, L. d'Orazio, Improving user interaction in mobile-cloud database query processing, in: 2017 IEEE International Conference on Big Data (Big Data), 2017, pp. 2500–2507.

[27] W. Kießling, Foundations of preferences in database systems, in: Proceedings of the 28th International Conference on Very Large Data Bases, VLDB Endowment, 2002, pp. 311–322.

[28] W. Kießling, G. Köstler, Preference sql: design, implementation, experiences, in: Proceedings of the 28th International Conference on Very Large Data Bases, VLDB Endowment, 2002, pp. 990–1001.

[29] R. Agrawal, R. Rantzau, E. Terzi, Context-sensitive ranking, in: Proceedings of the 2006 ACM SIGMOD International Conference on Management of data, ACM, 2006, pp. 383–394.

[30] B. Mohammed, M. Mouhoub, E. Alanazi, Combining Constrained CP-Nets and Quantitative Preferences for Online Shopping, Springer International Publishing 28th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE, 2015, pp. 702–711.

[31] F. Boubekeur, M. Boughanem, L. Tamine-Lechani, Towards flexible information retrieval based on cp-nets, in: Flexible Query Answering Systems, 7th International Conference, FQAS 2006, Milan, Italy, June 7-10, 2006, Proceedings, 2006, pp. 222–231, [Online]. Available: http://dx.doi.org/10.1007/11766254_19.

[32] F. Boubekeur, M. Boughanem, L. Tamine-Lechani, Semantic information retrieval based on cp-nets, in: FUZZ-IEEE 2007, IEEE International Conference on Fuzzy Systems, Imperial College, London, UK, 23-26 July, 2007, Proceedings, 2007, 2007, pp. 1–7, [Online]. Available: http://dx.doi.org/10.1109/FUZZY.2007.4295470.

[33] P. Ciaccia, Querying databases with incomplete cp-nets, in: Multidisciplinary Workshop on Advances in Preference Handling (M-PREF 2007), 2007.

[34] J. Rosati, T. Di Noia, T. Lukasiewicz, R. De Leone, A. Maurino, Preference Queries with Ceteris Paribus Semantics for Linked Data, On the Move to Meaningful Internet Systems: OTM 2015 Conferences: Confederated International Conferences:Springer International Publishing, Cham, 2015, pp. 423–442.

[35] H. Wang, X. Zhou, W. Chen, P. Ma, Top-k retrieval using conditional preference networks, in: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, in: ser. CIKM '12, ACM, New York, NY, USA, 2012, pp. 2075–2079, [Online]. Available: http://doi.acm.org/10.1145/2396761.2398576.

[36] I. Jolliffe, Principal Component Analysis, Wiley Online Library, 2005.

[37] L.M. Bruce, C.H. Koger, J. Li, Dimensionality reduction of hyperspectral data using discrete wavelet transform feature extraction, IEEE Trans. Geosci. Remote Sens. 40 (10) (2002) 2331–2338.

[38] H. Hosoya, A. Hyvärinen, Learning visual spatial pooling by strong pca dimension reduction, Neural Comput. 28 (7) (2016) 1249–1264, [Online]. Available: https://doi.org/10.1162/NECO_a_00843.

[39] J. Raigoza, J. Sun, Temporal join processing with hilbert curve space mapping, in: Proceedings of the 29th Annual ACM Symposium on Applied Computing, in: ser. SAC '14, ACM, New York, NY, USA, 2014, pp. 839–844, [Online]. Available: http://doi.acm.org/10.1145/2554850.2554903.

[40] A. Fisher, A new algorithm for generating hilbert curves, Softw. - Pract. Exp. 16 (1) (1986) 5–12.

[41] A. Cole, Direct transformations between sets of integers and hilbert polygons, Int. J. Comput. Math. 20 (2) (1986) 115–122.

[42] G. Jin, J. Mellor-Crummey, Sfcgen: A framework for efficient generation of multi-dimensional space-filling curves by recursion, ACM Trans. Math. Software 31 (1) (2005) 120–148.

[43] S.-I. Kamata, R.O. Eason, Y. Bandou, A new algorithm for n-dimensional hilbert scanning, IEEE Trans. Image Process. 8 (7) (1999) 964–973.

[44] A.R. Butz, Alternative algorithm for hilbert's space-filling curve, IEEE Trans. Comput. 100 (4) (1971) 424–426.

[45] J.K. Lawder, Calculation of mappings between one and n-dimensional values using the hilbert space-filling curve, School of Computer Science and Information Systems, Birkbeck College, University of London, London Research Report BBKCS-00-01 August, 2000.

[46] X. Zhao, Y. Lin, J. Heikkil?, Dynamic texture recognition using multiscale pca-learned filters, in: 2017 IEEE International Conference on Image Processing (ICIP), 2017, pp. 4152–4156.

[47] S. Lu, X. Wang, A new methodology to estimate the rotating phase of a bldc motor with its application in variable-speed bearing fault diagnosis, IEEE Trans. Power Electron. 33 (4) (2018) 3399–3410.

[48] S. Buchala, N. Davey, T.M. Gale, R.J. Frank, Analysis of linear and nonlinear dimensionality reduction methods for gender classification of face images, Int. J. Syst. Sci. 36 (14) (2005) 931–942, [Online]. Available: http://dx.doi.org/10.1080/00207720500381573.

[49] I.T. Jolliffe, Principal component analysis, pp. 1094–1096, 2011. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-04898-2_455.

[50] P. Demartines, J. Hérault, Curvilinear component analysis: a self-organizing neural network for nonlinear mapping of data sets, IEEE Trans. Neural Netw. 8 (1) (1997) 148–154, [Online]. Available: http://dx.doi.org/10.1109/72.554199.

[51] T.E. Allen, J. Goldsmith, H.E. Justice, N. Mattei, K. Raines, Generating cp-nets uniformly at random, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA, 2016, pp. 872–878, [Online]. Available: http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12439.

[52] R. Bayer, E.M. McCreight, Organization and maintenance of large ordered indexes, in: Pioneers and Their Contributions to Software Engineering, Springer, 2001, pp. 41–59.

[53] B. Desai, P. Goyal, F. Sadri, Composite index in ddbms, J. Syst. Softw. 8 (2) (1988) 105–119.

[54] K. Wu, E.J. Otoo, A. Shoshani, Optimizing bitmap indices with efficient compression, ACM Trans. Database Syst. 31 (1) (2006) 1–38.

[55] M. Morzy, T. Morzy, A. Nanopoulos, Y. Manolopoulos, Hierarchical bitmap index: An efficient and scalable indexing technique for set-valued attributes, in: Advances in Databases and Information Systems, Springer, 2003, pp. 236–252.

[56] K. Wu, A. Shoshani, K. Stockinger, Analyses of multi-level and multi-component compressed bitmap indexes, ACM Trans. Database Syst. 35 (1) (2010) 2.