Mining Order-Preserving SubMatrices from Probabilistic Matrices

QIONG FANG, Hong Kong University of Science and Technology
WILFRED NG, Hong Kong University of Science and Technology
JIANLIN FENG, Sun Yat-Sen University
YULIANG LI, Hong Kong University of Science and Technology

The *Order-Preserving SubMatrices* (OPSMs) capture consensus trends over columns shared by rows in a data matrix. Mining OPSM patterns discovers important and interesting local correlations in many real applications, such as those involving biological data or sensor data. The prevalence of uncertain data in various applications, however, poses new challenges for OPSM mining, since data uncertainty must be incorporated into OPSM modeling and the algorithmic aspects.

In this paper, we define new probabilistic matrix representations to model uncertain data with continuous distributions. A novel *Probabilistic Order-Preserving SubMatrix* (POPSM) model is formalized in order to capture similar local correlations in probabilistic matrices. The POPSM model adopts a new probabilistic support measure that evaluates the extent to which a row belongs to a POPSM pattern. Due to the intrinsic high computational complexity of the POPSM mining problem, we utilize the anti-monotonic property of the probabilistic support measure and propose an efficient Apriori-based mining framework called PROBAPRI to mine POPSM patterns. The framework consists of two mining methods, UNIAPRI and NORMAPRI, which are developed for mining POPSM patterns respectively from two representative types of probabilistic matrices, the *UniDist matrix* (assuming uniform data distributions) and the *NormDist matrix* (assuming normal data distributions). We show that the NORMAPRI method is practical enough for mining POPSM patterns from probabilistic matrices that model more general data distributions.

We demonstrate the superiority of our approach by two applications. First, we use two biological datasets to illustrate that the POPSM model better captures the characteristics of the expression levels of biologically correlated genes, and greatly promotes the discovery of patterns with high biological significance. Our result is significantly better than the counterpart OPSMRM (OPSM with Repeated Measurement) model which adopts a set-valued matrix representation to capture data uncertainty. Second, we run the experiments on an RFID trace dataset and show that our POPSM model is effective and efficient in capturing the common visiting subroutes among users.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications

General Terms: Algorithms

Additional Key Words and Phrases: order-preserving submatrices, probabilistic matrices, probabilistic support, OPSM mining

ACM Reference Format:
Qiong Fang, Wilfred Ng, Jianlin Feng, and Yuliang Li, 2014. Mining Order-Preserving SubMatrices from Probabilistic Matrices. *ACM Trans. Datab. Syst.* 39, 1, Article A (January 2014), 44 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

This work is partially supported by Hong Kong RGC grants under project number GRF 617610 and T12-403/11-2 and China NSF Grant 60970043.
Author's addresses: Q. Fang and W. Ng, Computer Science and Engineering Department, Hong Kong University of Science and Technology; J. Feng, School of Software, Sun Yat-Sen University; Y. Li, (Current address) Computer Science and Engineering Department, University of California, San Diego.

G \ T	t_1	t_2	t_3	t_4
g_1	10	**8**	**6**	**15**
g_2	8	**7**	**4**	**9**
g_3	2	**12**	**9**	**14**
g_4	6	10	12	7

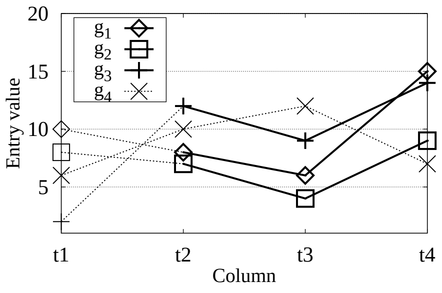


Fig. 1. An order-preserving submatrix (P, Q) with $P = \{g_1, g_2, g_3\}$, $Q = \{t_2, t_3, t_4\}$, and the consensus trend represented by $[t_3 \prec t_2 \prec t_4]$.

1. INTRODUCTION

Matrices are a common data presentation in many applications, such as those involving biological data [Seidel 2008] or sensor data [Rashidi et al. 2011; Gu et al. 2011]. For example, in bioinformatics, the DNA microarray data can be organized as a gene expression matrix, where rows correspond to genes, columns correspond to experimental conditions, and an entry stores the expression level of a gene measured under a certain condition. Various submatrix models have been proposed to capture different local correlations existing among a set of rows and a set of columns in a data matrix [Madeira and Oliveira 2004]. The *Order-Preserving SubMatrix* (OPSM) model [Ben-Dor et al. 2002], which is a well-known submatrix model, aims to capture the correlation that the entry values of a set of rows follow a consensus trend under a set of columns. An OPSM example is given in Figure 1. In the matrix shown in the left table, the submatrix with the row set $P = \{g_1, g_2, g_3\}$ and the column set $Q = \{t_2, t_3, t_4\}$ is an OPSM pattern, since, by ordering the entry values of every row in P under the columns in Q in increasing order, all the rows in P follow the same varying trend represented by the order $[t_3 \prec t_2 \prec t_4]$. The OPSM pattern is highlighted using real lines in the right diagram of Figure 1.

Given a data matrix, mining the submatrices that satisfy the OPSM model is called the OPSM mining problem. The OPSM patterns mined from different application data have led to the discovery of interesting knowledge, such as common functional ontologies of gene sets [Ben-Dor et al. 2002] and similar sequential behaviors shared by user groups [Pei et al. 2001; Rashidi et al. 2011]. Therefore, even having the complexity of NP-completeness [Ben-Dor et al. 2002], the OPSM mining problem has still been receiving a lot of research attention [Liu and Wang 2003; Gao et al. 2006; Chui et al. 2008; Zhang et al. 2008; Fang et al. 2010; Fang et al. 2012; Yip et al. 2013].

Currently, most of the work studies the OPSM mining problem under the assumption of having only deterministic data in matrices, such as the real-valued matrices and the set-valued matrices in which each entry stores a single or a set of real values. On the other hand, uncertain data are ubiquitous in many data-intensive applications [Ré et al. 2008; Li and Deshpande 2010; Soliman and Ilyas 2009; Aggarwal et al. 2009; Zhao et al. 2012; Muzammal and Raman 2011]. The prevalence of uncertain data in various applications, however, poses new challenges for OPSM mining, since it is nontrivial to incorporate data uncertainty into OPSM modeling and the algorithmic aspects. We now use the following two application examples to illustrate the importance of the OPSM model in capturing local correlations in uncertain data.

Table I. A UniDist Matrix $M^U(G,T)$ with $G = \{g_1, g_2, g_3\}$ and $T = \{t_1, t_2, t_3, t_4\}$

G \ T	t_1	t_2	t_3	t_4
g_1	$[7,8]$	$[6,7]$	$[2,4]$	$[10,12]$
g_2	$[4,8]$	$[7,8]$	$[3,5]$	$[14,15]$
g_3	$[4,8]$	$[3,5]$	$[3,7]$	$[9,11]$

Example 1.1. *In gene expression analysis, the expression levels of genes under different experimental conditions are monitored. Due to inevitable noise contamination, the experiment concerning a gene under a certain condition is usually repeated multiple times, and the noisy replicates are assumed to follow a Gaussian (or normal) distribution according to the studies in [Hughes et al. 2000; Lee et al. 2000]. Thus, to better describe the characteristics of such noisy gene expression data, each entry in the gene expression matrix, which corresponds to the expression level of a gene under a certain condition, should be associated with a normal distribution. Although the OPSM model is known to be effective for discovering biological correlations in real-valued gene expression matrices [Ben-Dor et al. 2002], it has to be adapted for the gene expression matrix having normally distributed probabilistic data.*

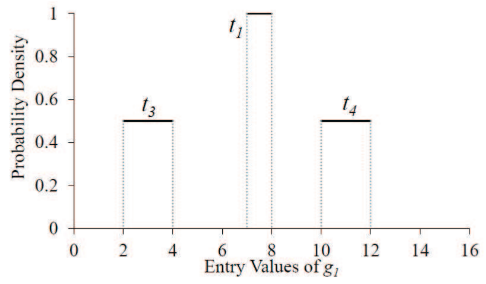
Example 1.2. *In order to track visitors' routes within a building, visitors are given RFID tags to wear and they are detected by RFID readers (or antennas) [Ré et al. 2008; Welbourne et al. 2008]. At an RFID detection site, a visitor may be continuously detected, which implies the probability of his / her stay at a particular location. Such RFID data can be organized as a probabilistic matrix where rows are visitors, columns are RFID detection sites, and each entry stores a time range indicating the stay of a visitor at a certain location. We are then able to find groups of visitors who likely share a common visiting subroute, which can be represented by a probabilistic OPSM model.*

To better model the data uncertainty as discussed in the above examples, we propose the *probabilistic matrices*, where each entry is associated with a continuous probabilistic distribution. Two commonly used distributions are the uniform and normal distributions. When the uniform distribution is assumed, we call such a probabilistic matrix a *UniDist matrix*. Table I shows an example of a UniDist matrix with 3 rows and 4 columns, denoted by $M^U(G,T)$. The entry $M^U(g_2, t_3)$ stores a range $[3, 5]$, which means that the corresponding entry value is uniformly distributed in the range between 3 and 5.

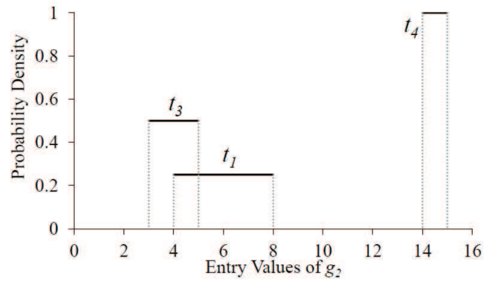
In scientific studies, the normal distribution is fundamental to modeling empirical data distributions. For example, it is recognized that the normal distribution is desirable for modeling the noisy gene expression data generated from the microarray experiments [Hughes et al. 2000; Lee et al. 2000; Nguyen et al. 2010]. When the normal distribution is considered, we call such a probabilistic matrix a *NormDist matrix* and denote it by $M^N(G,T)$. An entry $M^N(g_i, t_j)$ in a NormDist matrix is represented by a pair (μ_{ij}, σ_{ij}), where μ_{ij} and σ_{ij} are respectively the mean and standard deviation.

In this paper, we focus on tackling the OPSM mining problem based on two types of probabilistic matrices, the UniDist matrix and the NormDist matrix. However, we emphasize that the OPSM model defined based on the NormDist matrix and the corresponding mining method are flexible, in the sense that they can be adapted to dealing with probabilistic matrices having more general continuous distributions. This benefit will be further elaborated when the NormDist matrix is discussed.

Referring again to Figure 1, the submatrix (P, Q) is called an OPSM, if, for every row in P, its entry values under columns in Q induce an identical order of Q. The *induced*



(a) g_1 under Q with $\tau_Q = [t_3 \prec t_1 \prec t_4]$



(b) g_2 under Q with $\tau_Q = [t_3 \prec t_1 \prec t_4]$ and $\tau'_Q = [t_1 \prec t_3 \prec t_4]$

Fig. 2. Illustration of the order-preserving relationship in the UniDist matrix (g_1 and g_2 under $Q = \{t_1, t_3, t_4\}$ as shown in Table I). Given a range $[l, u]$, the probability density within the range is computed by $\frac{1}{u-l}$.

order of a row g under Q is acquired by sorting the numerical entry values of g under Q in increasing order, and then replacing the values by the corresponding column labels. However, the concept of induced order of the OPSM model is not applicable to probabilistic matrices.

Let us analyze the problem by referring to the UniDist matrix shown in Table I as an example. If the range entries of row g_1 under the columns $Q = \{t_1, t_3, t_4\}$ are arranged along the axis, we find that the values covered by the range $M^U(g_1, t_3)$ are smaller than the values covered by the range $M^U(g_1, t_1)$, and in addition, the values covered by both $M^U(g_1, t_3)$ and $M^U(g_1, t_1)$ are smaller than the values covered by the range $M^U(g_1, t_4)$, as illustrated in Figure 2(a). If the induced order of a row is determined by the relationship among values in the ranges, we can say that the induced order of row g_1 under Q is $\tau_Q = [t_3 \prec t_1 \prec t_4]$, or in other words, row g_1 supports the order τ_Q. However, if we arrange the entries of row g_2 under Q along the axis in a similar way, the entries $M^U(g_2, t_3)$ and $M^U(g_2, t_1)$ overlap on a subrange $[4, 5]$, as illustrated in Figure 2(b). Thus, g_2 is found to support both τ_Q and another possible order $\tau'_Q = [t_1 \prec t_3 \prec t_4]$. A similar problem also exists in the NormDist matrix. The fact that a row may support more than one order in the probabilistic matrices makes it necessary to evaluate the extent to which a row supports an order. Intuitively, in the above example, g_1 should be regarded as *better supporting* the order τ_Q than g_2 does. Motivated by this observation, we define a new measure, called *probabilistic support*, to evaluate the extent to which a row supports an order, or in other words, it is the probability that a row is likely to induce an order. Based on the probabilistic support, a new OPSM model called *Probabilistic OPSM* (POPSM) is defined.

Mining OPSM patterns from a real-valued matrix is an intrinsically difficult problem, which is proved to be NP-complete [Ben-Dor et al. 2002]. When mining POPSM patterns from the probabilistic matrices, we utilize the following strategies to design an efficient mining method. First, we prove the anti-monotonic property of the new probabilistic support measure, and make use of it to control the number of candidate patterns. Then, by combining a prefix-tree structure and the dynamic programming techniques, we are able to efficiently verify the candidate patterns and exhaustively mine all valid POPSM patterns from the UniDist matrix and the NormDist matrix.

In summary, we tackle a new problem of mining probabilistic OPSM patterns. The main contributions arising from this study are twofold.

— In the modeling aspect, we propose new probabilistic matrix representations to incorporate the data uncertainty commonly existing in many applications. Based on two types of probabilistic matrices, namely the UniDist matrix and the NormDist matrix, we define a new probabilistic OPSM model by adopting the probabilistic support measure that evaluates the extent to which a row is likely to induce an order. We compare our model with the counterpart *OPSM with Repeated Measurements* (OPSMRM) model [Chui et al. 2008; Yip et al. 2013], which is defined based on the set-valued matrices. We demonstrate the superiority of the POPSM model by running experiments on two real biological datasets and one RFID dataset.
Specifically, in the experiments using biological datasets, based on both the UniDist and NormDist matrices, the fraction of POPSM patterns that reach the highest *significance level* is larger than the fraction of the OPSMRM patterns that reach the same level, while the fraction of the POPSM patterns that fall in the lowest significance level is less than that of the OPSMRM patterns at the same level. Using the RFID trace data, we show that the common subroutes of a set of users can be accurately discovered with the adoption of the POPSM model.
— In the algorithmic aspect, we propose an efficient Apriori-based POPSM mining framework called PROBAPRI. There are two versions of PROBAPRI, denoted by UNIAPRI and NORMAPRI, which, respectively, mine POPSM patterns from the UniDist and NormDist matrices. PROBAPRI employs a *CandTree* structure to organize the POPSM patterns. Two dynamic programming (DP) techniques are developed for computing the probabilistic support. By interweaving the traversal of *CandTree* and the computation of the DP functions, the POPSM patterns can be efficiently verified during the mining process. NORMAPRI adopts the *spline* technique [Heath 2002] to approximate the normal distributions with simpler low-degree polynomials. The approximation step is general enough to deal with other continuous distributions. Thus, NORMAPRI is capable of mining POPSM patterns from more general probabilistic matrices.

The organization of the rest of this paper is as follows. We present the related work in Section 2. The notations and terminologies are introduced in Section 3. In Section 4, after introducing the probabilistic support measure, we define the *Probabilistic OPSM* (POPSM) model. The POPSM mining method PROAPRI is discussed in Section 5. Experiments on synthetic and real datasets are presented in Section 6. Finally, we conclude the paper in Section 7.

2. RELATED WORK

The problem of mining submatrix patterns was first studied over forty years ago for analyzing voting data [Hartigan 1972], where the voting data are organized as a matrix with voters as rows, candidates as columns and voting scores as entries. It is nearly impossible to find a group of voters who gave the same voting scores over all the candidates or to find a group of candidates who received the same voting scores from all the voters, especially when the number of voters and candidates is large. However, it is also observed that a subset of voters are likely to have common preferences over a subset of candidates, and such *local correlations* can be captured by submatrix patterns. As matrices become a common data representation in many application domains, the problem of mining submatrix patterns has been extensively studied for revealing various local correlations, which is usually known as subspace clustering [Parsons et al. 2004; Kriegel et al. 2009], co-clustering [Wang et al. 2011; Ding et al. 2006; Ji et al. 2012], and biclustering [Cheng and Church 2000; Madeira and Oliveira 2004]. We now give more details of the work.

2.1. Subspace Clustering, Co-Clustering, and Biclustering

Studies about subspace clustering are motivated by the observation that the data points which are irrelevant in high dimensional space may be well clustered or correlated in lower dimensional subspaces [Parsons et al. 2004; Kriegel et al. 2009]. These studies aim to identify a subset of attributes or features that form a subspace, and find the data points that form a cluster over the subspace. If high-dimensional data are organized as a matrix, the subspace features together with the clustered data points correspond to the submatrix patterns in the matrix. However, different from the order-preserving relationship required by the OPSM model, the criteria for subspace clustering are usually the Euclidean distance between data points [Agrawal et al. 1998; Aggarwal et al. 1999; Moise and Sander 2008], the linearity correlation among data points [Günnemann et al. 2012], the density of clusters [Kailing et al. 2004], or the statistical significance of clusters [Moise and Sander 2008].

Another stream of submatrix pattern mining work aims to partition a matrix into grid-distributed disjoint submatrices such that the subset of rows and the subset of columns in each individual submatrix are expected to be highly correlated with each other. This category of problems is usually called *co-clustering*, and has mainly been studied in recommender systems and text mining [Daruru et al. 2009; Banerjee et al. 2004; Dhillon et al. 2003; Pan et al. 2008; Long et al. 2005; Ji et al. 2012]. The grid distribution structure of submatrix patterns required by the co-clustering methods avoids the explosion of the number of mined patterns, which however is rather restrictive. In order to achieve a grid structure that optimizes the overall pattern scores, the quality of a part of submatrix patterns usually has to be sacrificed.

Cheng and Church [Cheng and Church 2000] first adopted the submatrix mining methods to analyze the biological gene expression data and called the problem *biclustering*. Biclustering methods aim to formulate different submatrix models so as to capture the biological correlations among a subset of genes and a subset of conditions. Madeira et al. [Madeira and Oliveira 2004] classified existing submatrix models into four categories: *submatrix with constant values* [Busygin et al. 2002], *submatrix with constant rows or constant columns* [Getz et al. 2000; Pandey et al. 2009; Gupta et al. 2010], *submatrix with coherent values* [Cho et al. 2004; Pontes et al. 2010], and *submatrix with coherent evolutions* [Murali and Kasif 2003; Tanay et al. 2002; Gupta and Aggarwal 2010; Li et al. 2009; Madeira et al. 2010]. The OPSM model belongs to the fourth category according to this classification.

2.2. Mining Order-Preserving Submatrices

The *Order-Preserving Submatrix* (OPSM) model, proposed by Ben-Dor et al. [Ben-Dor et al. 2002], aims to capture the fact that the entry values of a set of rows exhibit the same trend under a set of columns. A comparative study conducted by Prelić et al. [Prelić et al. 2006] showed that, compared to five other coherent-value or coherent-evolution submatrix models [Cheng and Church 2000; Prelić et al. 2006; Tanay et al. 2002; Murali and Kasif 2003; Ihmels et al. 2002], the OPSM model better captures the association of correlated genes and conditions and promotes the discovery of a larger fraction of biologically significant patterns. However, it is also recognized that the OPSM model may be too strict to be practical, since real gene expression data are noisy and the identical trend is usually hard to preserve [Ben-Dor et al. 2002]. To address this problem, various noise-tolerant OPSM models have been proposed in literature, such as the *Approximate Order-Preserving Cluster* (AOPC) model [Zhang et al. 2008], the *Relaxed Order-Preserving SubMatrix* (ROPSM) model [Fang et al. 2010], the *Bucket Order-Preserving SubMatrix* (BOPSM) model [Fang et al. 2012], the *Generalized BOPSM* (GeBOPSM) model [Fang et al. 2012], and the *error-tolerated OPSM* model [Cheung et al. 2007]. The AOPC model relaxes the condition that all the rows in an OPSM should induce the same linear order of columns, and only requires a pre-specified fraction of rows to induce the same linear order. The ROPSM model further relaxes the AOPC model, and allows all the rows in an ROPSM pattern to induce orders similar to the backbone order of the pattern. The BOPSM model tries to capture the consensus staged trend of a set of rows over a set of columns, and requires every row in a BOPSM pattern to induce an identical bucket order (i.e., an order of sets). The GeBOPSM model is a generalization of AOPC, ROPSM, and BOPSM. It allows the rows in a GeBOPSM pattern to induce bucket orders which are similar to the backbone order of the GeBOPSM pattern. The error-tolerated OPSM model still requires all the rows in a pattern to induce an identical order of a set of columns but allows the entry values of a row under two adjacent columns to violate the ordering relationship within a pre-specified error threshold. However, due to different scalings of expression values of different genes, it is difficult to set a proper absolute error threshold to guarantee that all the rows in a pattern still follow a consensus varying trend.

While the OPSM model and several other noise-tolerant OPSM models are all defined based on real-valued data matrices, an alternative way to address the issue of noisy data is to keep a set of replicates for each entry in the matrix, and such data matrices can be presented as *set-valued matrices* [Hughes et al. 2000; Nguyen et al. 2010; Ideker et al. 2001]. In a set-valued matrix, it is actually assumed that the set of replicates in every entry are equally likely to be observed. If an entry stores a set of k replicates, all of them have an equal probability of $\frac{1}{k}$. Therefore, a set-valued matrix can be regarded as a probabilistic matrix with discrete distribution. Based on the set-valued matrices, an *OPSM with Repeated Measurement* (OPSMRM) model [Chui et al. 2008; Yip et al. 2013] was introduced, where a fractional support measure is adopted to evaluate the extent to which a row supports a linear order. However, when the number of replicates is small, as the noisy and true replicates have equally large probabilities, the fractional support is easily affected by one or two noisy replicates. On the other hand, when the number of replicates grows, the cost of computing the fractional support increases sharply, which greatly degrades the performance of mining OPSMRM patterns.

Continuous distributions such as normal distribution are known to be effective for smoothing out the influence of noise in scientific experiments, and thus are commonly adopted to infer the error model of scientific data. For example, the observational error in gene expression analysis, which may be caused by instrumental limits or measurement errors, is assumed to follow a normal distribution [Hughes et al. 2000; Nguyen et al. 2010; Chia and Karuturi 2010]. Thus, a gene expression matrix can be presented as a probabilistic matrix with normal distribution. Probabilistic matrices are also a natural representation of the data arising from many sensor applications such as RFID tracking systems [Ré et al. 2008]. For example, when a user is detected at a particular RFID detection site within a time range, such trace data can then be represented as a probabilistic matrix with uniform distribution. The POPSM model proposed in this paper is defined based on such probabilistic matrices with continuous distributions. We summarize the characteristics of the OPSM model and its variants in Table II.

Ben-Dor et al. proved that mining OPSM patterns is an NP-complete problem [Ben-Dor et al. 2002]. Then, they proposed a model-based method which aims to mine the best OPSM in terms of the statistical significance, since patterns with high statistical significance are regarded more likely to be biologically significant. Their method keeps a limited number of partial models which are smaller OPSM patterns. Then, it expands the partial models into larger and thus more statistically significant patterns. Their method, however, is heuristic-based, and the significance of the mined OPSM patterns is very sensitive to the selection of the partial models. Trapp et al.

Table II. A Summary of OPSM Related Models

Models	Matrix Types	Pattern Characteristics	References
OPSM Twig OPSM	Real-valued matrices	Strict order-preserving	[Ben-Dor et al. 2002] [Gao et al. 2006]
AOPC ROPSM BOPSM, GeBOPSM Error-tolerated OPSM	Real-valued matrices	Relaxed order-preserving	[Zhang et al. 2008] [Fang et al. 2010] [Fang et al. 2012] [Cheung et al. 2007]
OPSMRM	Set-valued matrices	Fractional support to order	[Chui et al. 2008] [Yip et al. 2013]
POPSM	**Probabilistic matrices with continuous distributions**	**Probabilistic support to order**	**This work**

[Trapp and Prokopyev 2010] and Humrich et al. [Humrich et al. 2011] both made use of integer programming techniques and proposed similar methods for mining the maximal OPSM pattern. While Ben-Dor et al., Trapp et al., and Humrich et al.'s work all aimed to mine a single optimal OPSM pattern, Liu et al. [Liu and Wang 2003] proposed a tree-based OPSM mining method, called *OPC-Tree*, to exhaustively mine all the OPSM patterns that satisfy some size thresholds. However, when the number of columns in the data matrix increases, the size of the tree grows extremely large, and the performance of pattern mining is greatly degraded. Cheung et al. [Cheung et al. 2007] adopted the sequential pattern mining method in [Agrawal and Srikant 1995; Srikant and Agrawal 1996], and developed an Apriori-based method to exhaustively mine OPSM patterns that satisfy some size thresholds. A post-processing solution was additionally designed to combine mined OPSM patterns into error-tolerated OPSM patterns [Cheung et al. 2007]. Gao et al. proposed a KiWi framework in [Gao et al. 2006; Gao et al. 2012] to mine *twig OPSM* patterns, which contain a large number of columns and very few rows. The framework expands a limited number of linear orders of columns in a breadth-first manner, and applies very strong conditions for pruning those linear orders that are less likely to grow into twig OPSM patterns. Their method is shown to be efficient but valid twig OPSM patterns may also get pruned. Zhang et al. [Zhang et al. 2008] adopted a similar pattern merging strategy as in [Cheung et al. 2007] to mine AOPC patterns. Taking a set of OPSM patterns as input, The AOPC mining method merges pairs of OPSM patterns into AOPC patterns in a greedy way until no more AOPC patterns can be generated. Fang et al. [Fang et al. 2010] proposed a pattern growth method, which expands seed OPSM patterns into ROPSM patterns. Later, they developed an Apriori-based method for mining BOPSM patterns and utilized the anti-monotonic property of the BOPSM model to control candidate generation [Fang et al. 2012]. However, the BOPSM model is defined based on the real-valued matrices and the rows in the BOPSM patterns are assumed to induce consensus bucket orders. Thus, the proof of the anti-monotonic property of the BOPSM model is different from that of the anti-monotonic property of the POPSM model. An efficient prefix-tree structure was designed to maintain the BOPSM patterns, which motivates us to employ the *CandTree* to compactly organize the POPSM patterns in this work.

The OPSMRM mining method [Chui et al. 2008; Yip et al. 2013] also follows the Apriori-based framework, and consists of the following two steps. First, an Apriori-based method is taken to exhaustively mine frequent orders of columns. An order is regarded frequent if the sum of fractional supports contributed by all the rows exceeds a certain threshold. In this step, the anti-monotonic property of the fractional support measure is made use of. However, as the fractional support is defined based on the set-valued matrices, its computation method and the proof of the anti-monotonic property are different from our work that involves the notion of probabilistic support. Then, for each frequent order, the set of rows whose fractional supports to it satisfy another inclusion threshold are picked. The selected rows, together with the columns involved in the order, form an OPSMRM pattern. The inclusion threshold plays the similar role as the support threshold in our POPSM model. The OPSMRM mining process, however, has the following implications. In the first step, although the fractional support of every row with respect to an order is small, this order may still be regarded as frequent due to a large enough sum of the fractional supports contributed by all the rows. Then in the second step, a frequent order may fail to lead to a valid OPSMRM pattern if none of the rows has a large enough fractional support that satisfies the inclusion threshold. Another possibility is that, very few rows have large enough fractional supports with respect to the frequent order, and this order finally leads to a very small and hence statistically insignificant patterns [Ben-Dor et al. 2002].

2.3. Sequential Pattern Mining

If a data matrix is transformed into a set of attribute (i.e., column label) sequences ordered by their values in every row, the matrix can be viewed as a transaction database with a collection of sequences. Accordingly, the OPSM mining problem is converted to a frequent sequential pattern mining problem [Agrawal and Srikant 1995; Srikant and Agrawal 1996].

However, due to some unique properties of the OPSM mining problem, using frequent sequential pattern mining methods for mining OPSM patterns is not satisfactory from the efficiency view point. First, each attribute appears at most once in each sequence. Second, as the data matrices in an OPSM mining application are usually very dense, the transformed sequences are also dense, in the sense that every attribute may appear in most of the sequences. As a result, the searching space of the depth-first sequential pattern mining methods would be extremely large, while few candidates can be pruned in the first few rounds of the breadth-first pattern mining methods. Liu et al. [Liu and Wang 2003] took into account the characteristics of the OPSM mining problem, and proposed an OPSM mining method called *OPC-Tree*, which improves the basic techniques of a well-known efficient frequent sequential pattern mining method called PrefixSpan [Pei et al. 2001; Pei et al. 2004]. The *OPC-Tree* was shown to outperform PrefixSpan in mining OPSMs. Agrawal et al. proposed an Apriori-based sequential mining method [Agrawal and Srikant 1995; Srikant and Agrawal 1996], based on which the BOPSM mining method [Fang et al. 2012] and the OPSMRM mining method [Chui et al. 2008] were respectively developed. The PROBAPRI method proposed in this work also adopts the Apriori-based framework, but we apply it to dealing with the probabilistic data with continuous distributions. Kum et al. [Kum et al. 2003] studied the problem of mining approximate frequent sequential patterns. They first clustered input sequences into disjoint sets, and then looked for consensus patterns within each cluster. However, unlike the support requirement for mining OPSM patterns, the consensus patterns were evaluated based on their global support, and thus a frequent consensus pattern is not necessarily supported by a large enough number of sequences.

Aggarwal et al. [Aggarwal et al. 2009], Muzammal et al. [Muzammal and Raman 2011] and Zhao et al. [Zhao et al. 2012] studied the problem of mining frequent patterns or frequent sequential patterns in the context of uncertain data. However, the uncertain data are modeled as discrete values rather than continuous distributions.

2.4. Modeling Probabilistic Data with Continuous Distributions

The uncertain data in [Li and Deshpande 2010; Soliman and Ilyas 2009] are modeled using continuous distributions. But their work are related to the problem of top-k query processing and ranking in uncertain databases. We are essentially studying the submatrix pattern mining problem. In Soliman et al.'s work, a positional probability is needed to be computed, and the formulation of the probability is similar to that of our probabilistic support. However, they only considered the uniform distribution. The Monte-Carlo integration method proposed for computing the positional probability only returns an approximate answer. In Li et al.'s work, the spline technique was also adopted to approximate complex continuous distributions with piecewise polynomials. However, the techniques they proposed to compute a parameterized ranking function is totally different from the techniques we proposed for mining POPSM patterns.

Li et al. also discussed two other approximation techniques for continuous distributions, the *Monte Carlo* method and the *discretization* method. The Monte Carlo method approximates a continuous distribution by a set of independent random samples drawn from the distribution, which converts a probabilistic matrix to a set-valued matrix. This approximation is not applicable for mining POPSM patterns. The discretization method approximates a continuous distribution by piecewise histograms, which can be perceived as a specialization of the spline approximation.

3. PRELIMINARIES

In this section, we introduce some notations and terminologies that are used throughout the paper.

3.1. UniDist Matrix and Range

We denote an m-by-n UniDist matrix by $M^U(G, T)$, where G is a set of m rows and T is a set of n columns (or items). The entry $M^U(g_i, t_j)$ under row g_i and column t_j is represented by a range $R_{ij} = [l_{ij}, u_{ij}]$, where l_{ij} and u_{ij} are respectively the lower and upper bounds of the range. We call the set of range entries of a row g_i under the columns in T, i.e., $\{[l_{i1}, u_{i1}], \ldots, [l_{in}, u_{in}]\}$, the *record of* g_i.

For each entry $M^U(g_i, t_j)$, its possible replicates are assumed to be uniformly distributed within the range, and thus it can also be denoted by a random variable M^U_{ij} with its *probability density function* (PDF) given by

$$p^U_{ij}(x) = \begin{cases} \frac{1}{u_{ij}-l_{ij}} & \text{if } x \in [l_{ij}, u_{ij}] \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

For example, the entry $M^U(g_1, t_3)$ in the UniDist matrix shown in Table I is $[2, 4]$. Thus, the corresponding random variable M^U_{13} has the probability density of $\frac{1}{4-2} = 0.5$ within the range $[2, 4]$ and 0 out of the range. When there is no ambiguity, we may use the notations $M^U(g_i, t_j)$, R_{ij}, $[l_{ij}, u_{ij}]$, and M^U_{ij} interchangeably to represent an entry in a given UniDist matrix.

In many real-life applications, the entry values in the data matrix are collected independently. For example, when the DNA microarray technology is adopted for gene expression analysis, the expression level of every gene under every condition is monitored and measured independently [Seidel 2008]. Therefore, the entries in the generated gene expression matrix can be regarded as independent of each other. In the RFID applications, an RFID reader is supposed to detect different RFID tags separately and two RFID readers do not influence each other [Welbourne et al. 2008]. Thus, the independence assumption also holds in the RFID data matrix. We now assume that the random variables M^U_{ij}, with $1 \le i \le m$ and $1 \le j \le n$, be independent of each other.

Given a set of ranges $\{[l_1, u_1], \ldots, [l_k, u_k]\}$, we can sort them in increasing order of their middle point values given by $\frac{l_i+u_i}{2}$, and get an ordered range set $\langle [l_{i_1}, u_{i_1}], \ldots, [l_{i_k}, u_{i_k}] \rangle$, where $\langle i_1, \ldots, i_k \rangle$ is a permutation of the values 1 to k. We say that the range $R_i = [l_i, u_i]$ is *smaller than* (or *larger than*) the range $R_j = [l_j, u_j]$ if $\frac{l_i+u_i}{2}$ is smaller than (or larger than) $\frac{l_j+u_j}{2}$. The *difference* between R_i and R_j, denoted by $D(R_i, R_j)$, is given by

$$D(R_i, R_j) = \left|\frac{l_i + u_i}{2} - \frac{l_j + u_j}{2}\right|.$$

We say that two ranges $R_i = [l_i, u_i]$ and $R_j = [l_j, u_j]$ are *disjoint* if $u_i < l_j$ or $u_j < l_i$. Notably, the fact that R_i is smaller than (or larger than) R_j does not necessarily mean that R_i and R_j are disjoint.

3.2. NormDist Matrix

Similarly to $M^U(G, T)$, we denote a NormDist matrix by $M^N(G, T)$. The entry $M^N(g_i, t_j)$ is represented by a normal distribution (μ_{ij}, σ_{ij}), where μ_{ij} and σ_{ij} are respectively the mean and the standard deviation. Similarly, we call the set of entries of a row g_i under the columns in T, i.e., $\{(\mu_{i1}, \sigma_{i1}), \ldots, (\mu_{in}, \sigma_{in})\}$, the *record of* g_i. In addition, an entry $M^N(g_i, t_j)$ can be denoted by a random variable M^N_{ij} with its PDF given by

$$p^N_{ij}(x) = \frac{1}{\sqrt{2\pi\sigma^2_{ij}}} e^{-\frac{(x-\mu_{ij})^2}{2\sigma^2_{ij}}}. \tag{2}$$

Similarly to M^U_{ij}, the random variables in a NormDist matrix, M^N_{ij} with $1 \le i \le m$ and $1 \le j \le n$, are also regarded as mutually independent. When there is no ambiguity, we use the notations $M^N(g_i, t_j)$, (μ_{ij}, σ_{ij}), and M^N_{ij} interchangeably to represent an entry.

Given two random variables M^N_1 and M^N_2 with the normal distributions (μ_1, σ_1) and (μ_2, σ_2), we say that M^N_1 is *smaller than* (or larger than) M^N_2 if $\mu_1 < \mu_2$ (or $\mu_1 > \mu_2$). The *difference* between M^N_1 and M^N_2, denoted by $D(M^N_1, M^N_2)$, is given by

$$D(M^N_1, M^N_2) = |\mu_1 - \mu_2|.$$

3.3. Probabilistic Matrices

When there is no ambiguity, we collectively call both the UniDist matrix and the NormDist matrix as the probabilistic matrices and use $M(G, T)$ to denote such matrices. The entry $M(g_i, t_j)$ with $g_i \in G$ and $t_j \in T$ is also denoted by M_{ij}.

3.4. Order

Given a set of k items $Q = \{t_1, \ldots, t_k\}$, a *linear order* (or simply an *order*) of Q is represented as $\tau_Q = [t_{i_1} \prec t_{i_2} \prec \cdots \prec t_{i_k}]$, and the ordering relation " $\prec$ " satisfies the criteria of *antisymmetry*, *transitivity*, and *linearity*. Q is called the *associated item set* of τ_Q, and τ_Q is said to be a *size-k order*.

Given two orders τ_1 and τ_2 with their associated item sets Q_1 and Q_2, we say that τ_1 is a *sub-order* of τ_2, if $Q_1 \subseteq Q_2$, and for all pairs of items $t_i, t_j \in Q_1$, $(t_i \prec t_j) \in \tau_1$ implies $(t_i \prec t_j) \in \tau_2$.

4. THE PROBABILISTIC OPSM MODEL

In this section, we first introduce a new measure, called *probabilistic support*, to evaluate the extent to which a row probably induces (or supports) an order. We then incor-

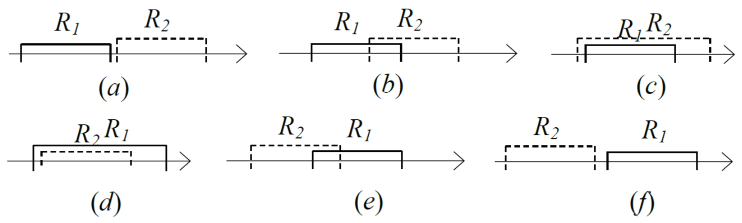


Fig. 3. Relative positions between two ranges R_1 and R_2

porate this measure into a new OPSM model called *Probabilistic OPSM* (or POPSM for short).

4.1. Probabilistic Support Measure

Definition 4.1 (*Probabilistic Support*). Given a row g_i and a set of columns Q in a probabilistic matrix M, the probabilistic support of g_i with respect to (w.r.t.) $\tau_Q = [t_1 \prec \cdots \prec t_r]$, denoted by $\mathcal{PS}(g_i, \tau_Q)$, is given by

$$\mathcal{PS}(g_i, \tau_Q) = P(M_{i1} < M_{i2} < \cdots < M_{ir}),$$

where M_{ij} is the random variable corresponding to the entry $M(g_i, t_j)$ with $1 \le j \le r$, and $P(M_{i1} < \cdots < M_{ir})$ is the probability that the random event $(M_{i1} < \cdots < M_{ir})$ occurs.

Based on the theory of order statistics [Ahsanullah et al. 2013], given a set of random variables $M_1, M_2, \ldots, M_r$ with the corresponding PDFs as $p_1(x)$, $p_2(x)$, …, $p_r(x)$, the probability of the random event $(M_1 < \cdots < M_r)$ is given by

$$P(M_1 < \cdots < M_r) = \int_{-\infty}^{+\infty} p_1(x_1)dx_1 \prod_{i=2}^{r} \left[\int_{x_{i-1}}^{+\infty} p_i(x_i)dx_i \right]. \quad (3)$$

In a UniDist matrix, the PDF of a random variable is given in Formula (1), and all the random variables are independent of each other as we have illustrated in Section 3. By replacing the PDF $p_i(x)$ of M_i with $\frac{I(x \in R_i)}{u_i - l_i}$, Formula (3) is transformed into the following expression:

$$\begin{aligned} P(M_1 < \cdots < M_r) &= \int_{-\infty}^{+\infty} \frac{I(x_1 \in R_1)}{u_1 - l_1} dx_1 \prod_{i=2}^{r} \left[\int_{x_{i-1}}^{+\infty} \frac{I(x_i \in R_i)}{u_i - l_i} dx_i \right], \\ &= \frac{\int_{-\infty}^{+\infty} I(x_1 \in R_1)dx_1 \prod_{i=2}^{r} \left[\int_{x_{i-1}}^{+\infty} I(x_i \in R_i)dx_i \right]}{\prod_{i=1}^{r}(u_i - l_i)}, \end{aligned} \quad (4)$$

where $R_i = [l_i, u_i]$ is the range corresponding to M_i and $I(\cdot)$ is an indicator function.

We now use two random variables M_1 and M_2 with uniform distribution to illustrate how the probability of the random event $(M_1 < M_2)$ varies with the relative position of the two corresponding ranges. Suppose that the associated ranges of M_1 and M_2 are respectively $R_1 = [l_1, u_1]$ and $R_2 = [l_2, u_2]$, and the range sizes $(u_1 - l_1)$ and $(u_2 - l_2)$ are fixed. There are altogether six possible relative positions between R_1 and R_2 as shown in Figure 3.

— Case (a): R_1 is smaller than R_2, and they are disjoint. In this case, the probability is 1.0.

— Case (b): R_1 is smaller than R_2, but they overlap each other. According to Formula (4), the probability is given by

$$1 - \frac{(u_1 - l_2)^2}{2(u_1 - l_1)(u_2 - l_2)}.$$

There are three sub-cases concerning the varying trend of the probability.

- If $(u_1 - l_1) < (u_2 - l_2)$, as $D(R_1, R_2)$ decreases from $\frac{(u_2-l_1)}{2}$ to $\frac{(u_2-u_1)}{2}$, the probability monotonically decreases from 1 to $\left(1 - \frac{u_1-l_1}{2(u_2-l_2)}\right)$;
- if $(u_1 - l_1) > (u_2 - l_2)$, as $D(R_1, R_2)$ decreases from $\frac{(u_2-l_1)}{2}$ to $\frac{(l_2-l_1)}{2}$, the probability monotonically decreases from 1 to $\left(1 - \frac{u_2-l_2}{2(u_1-l_1)}\right)$;
- if $(u_1 - l_1) = (u_2 - l_2)$, as $D(R_1, R_2)$ decreases from $\frac{(u_2-l_1)}{2}$ to 0, the probability monotonically decreases from 1 to $\frac{1}{2}$.

For all three sub-cases, the probability monotonically decreases as $D(R_1, R_2)$ gets smaller.

— Case (c): R_2 contains R_1. In this case, the probability falls in the range

$$\left[\frac{u_1 - l_1}{2(u_2 - l_2)},\ 1 - \frac{u_1 - l_1}{2(u_2 - l_2)}\right].$$

When $D(R_1, R_2)$ equals to 0, the probability equals to $\frac{1}{2}$.

— Case (d): R_1 contains R_2. This case is symmetric with Case (c), and the probability falls in the range

$$\left[\frac{u_2 - l_2}{2(u_1 - l_1)},\ 1 - \frac{u_2 - l_2}{2(u_1 - l_1)}\right].$$

When $D(R_1, R_2)$ equals 0, the probability equals $\frac{1}{2}$.

— Case (e): R_1 is larger than R_2, and they overlap each other. In this case, the probability is given by

$$\frac{(u_2 - l_1)^2}{2(u_1 - l_1)(u_2 - l_2)}.$$

Similarly to Case (b), there are also three sub-cases concerning the varying trend of the probability.

- If $(u_1 - l_1) < (u_2 - l_2)$, as $D(R_1, R_2)$ increases from $\frac{(l_1-l_2)}{2}$ to $\frac{(u_1-l_2)}{2}$, the probability monotonically decreases from $\frac{u_1-l_1}{2(u_2-l_2)}$ to 0;
- if $(u_1 - l_1) > (u_2 - l_2)$, as $D(R_1, R_2)$ increases from $\frac{(u_1-u_2)}{2}$ to $\frac{(u_1-l_2)}{2}$, the probability monotonically decreases from $\frac{u_2-l_2}{2(u_1-l_1)}$ to 0;
- if $(u_1 - l_1) = (u_2 - l_2)$, as $D(R_1, R_2)$ increases from 0 to $\frac{(u_1-l_2)}{2}$, the probability monotonically decreases from $\frac{1}{2}$ to 0.

For all three sub-cases, the probability monotonically decreases as $D(R_1, R_2)$ gets larger.

— Case (f): R_1 is larger than R_2 and they are disjoint. In this case, the probability equals 0.

Considering the probability $P(M_1 < M_2)$, where M_1 and M_2 are random variables that follow normal distribution, we can observe the varying trends similar to that of uniform distribution. In other words, when M_1 is smaller than M_2, the probability is larger than 0.5, and monotonically decreases from 1 to 0.5 as $D(M_1, M_2)$ gets smaller.

When M_1 is larger than M_2, the probability is smaller than 0.5 and monotonically decreases from 0.5 to 0 as $D(M_1, M_2)$ gets larger. Strictly speaking, the domain of a normal distribution is infinite, and thus the probability only infinitely approaches 1 when M_1 is smaller than M_2 and $D(M_1, M_2)$ gets extremely large. At the other extreme, the probability infinitely approaches 0 when M_2 is smaller than M_1 and $D(M_1, M_2)$ gets extremely large.

From the above analysis, Formula (3) well reflects the positional relationship and the difference between random variables. Thus, we adopt Formula (3) to formulate our probabilistic support measure.

The probabilistic support holds the anti-monotonic property, which is formally stated in Theorem 4.2.

THEOREM 4.2. *[Anti-Monotonicity] Given a threshold α, if the probabilistic support of a row g w.r.t. an order τ_Q is larger than or equal to α, the probabilistic support of g w.r.t. all the sub-orders of τ_Q is also larger than or equal to α.*

Proof: Let g be a row and $\{t_1, \dots, t_k\}$ be a set of k columns in a probabilistic matrix. Suppose the entries of g under $\{t_1, \dots, t_k\}$ correspond to the random variables $\{M_1, \dots, M_k\}$. To establish the proof, we only need to show that

$$P(M_1 < \cdots < M_k) \leq P(M_1 < \cdots < M_{r-1} < M_{r+1} < \cdots < M_k),$$

with $1 \leq r \leq k$, which is straightforward.

Since the occurrence of the random event $(M_1 < \cdots < M_k)$ implies the occurrence of the event $(M_1 < \cdots < M_{r-1} < M_{r+1} < \cdots < M_k)$, the probability $P(M_1 < \cdots < M_k)$ is thus no larger than the probability $P(M_1 < \cdots < M_{r-1} < M_{r+1} < \cdots < M_k)$. □

4.2. The POPSM Model

Using the probabilistic support measure, we now formalize the probabilistic OPSM model.

Definition 4.3 (*Probabilistic OPSM (POPSM)*). Given a probabilistic matrix $M(G, T)$ and a support threshold α, a submatrix (P, Q) with $P \subseteq G$ and $Q \subseteq T$ is said to be a *POPSM pattern*, if there exists an order τ_Q such that, for all $g_i \in P$, the probabilistic support g_i w.r.t. τ_Q is larger than or equal to α, that is,

$$\forall g_i \in P, \mathcal{PS}(g_i, \tau_Q) \geq \alpha.$$

We call the order τ_Q the *backbone order* of the POPSM (P, Q), and the rows in P the *supporting rows* of τ_Q.

Since a POPSM pattern (P, Q) is associated with a backbone order τ_Q, we denote such a pattern by $(P, Q : \tau_Q)$. A POPSM pattern $(P, Q : \tau_Q)$ is said to be *maximal*, if there does not exist any other POPSM pattern $(P', Q' : \tau_{Q'})$ such that $P \subseteq P'$, $Q \subseteq Q'$, and τ_Q is a sub-order of $\tau_{Q'}$.

Based on the anti-monotonic property of the probabilistic support, we are able to straightforwardly deduce that the POPSM model also satisfies the anti-monotonic property. That means, given a POPSM pattern $(P, Q : \tau_Q)$, all the submatrices (P', Q') of (P, Q) are also POPSM patterns with the backbone order $\tau_{Q'}$ being a sub-order of τ_Q. We will make use of the anti-monotonic property of the POPSM model to develop an efficient POPSM mining algorithm in Section 5.

Now, we formally define the POPSM mining problem.

Definition 4.4 (*The POPSM Mining Problem*). Given a probabilistic matrix $M(G, T)$, a support threshold α, and two size thresholds r_{min} and c_{min}, we aim to

ALGORITHM 1: PROBAPRI

Input: Probabilistic matrix $M(G, T)$; support threshold α; size thresholds r_{min} and c_{min}
Variable: $\mathcal{C}_k$ - the set of size-k candidate orders; $\mathcal{F}_k$ - the set of size-k frequent orders

```
$\mathcal{F}_1$ ={size-1 frequent orders};
for ($k = 2; \mathcal{F}_{k-1} \neq \phi; k++$) do
    $\mathcal{C}_k$= GENCAND($\mathcal{F}_{k-1}$) ;
    COUNTSUP($M, \mathcal{C}_k, \alpha$);
    $\mathcal{F}_k = \{\tau \mid \tau \in \mathcal{C}_k, supp(\tau) \geq r_{min}\}$ ;
    if $k \geq c_{min}$ && $\mathcal{F}_k \neq \phi$ then
        Put POPSM patterns in output pool;
    end
end
Output maximal POPSM patterns;
```

exhaustively mine from M all the maximal POPSM patterns that contain at least r_{min} rows and c_{min} columns.

As very small patterns possibly exist in the matrix randomly and they are likely to reveal trivial biological correlations [Ben-Dor et al. 2002; Liu and Wang 2003; Fang et al. 2010], we include two size thresholds r_{min} and c_{min} in Definition 4.4 to avoid mining too many small POPSM patterns.

5. MINING PROBABILISTIC OPSM

In this section, we exploit the anti-monotonic property of the POPSM model and adopt an Apriori-based framework to develop a new POPSM mining method called PROBAPRI. There are two versions of PROBAPRI, UNIAPRI and NORMAPRI, respectively developed for mining POPSM patterns from the UniDist matrix and the NormDist matrix. If not explicitly specified, the procedures presented are applicable to both UNIAPRI and NORMAPRI.

5.1. PROBAPRI Algorithm

Given a probabilistic matrix $M(G, T)$ and a support threshold α, a naïve way to mine POPSM patterns can be carried out as follows: for every set of columns Q with $Q \subseteq T$ and $|Q| \geq c_{min}$, and every possible order τ_Q of Q, we simply check all the rows in G to examine if the probabilistic support of a row w.r.t. τ_Q is larger than or equal to α. Let P be the set of supporting rows of τ_Q. If $|P| \geq r_{min}$, then (P, Q) is a valid POPSM pattern with τ_Q as the backbone order. However, such an exhaustive search is apparently infeasible, since the number of such orders is prohibitively large, especially when the number of columns in T is large.

The anti-monotonic property of the POPSM model is important in developing a more efficient approach for mining POPSM patterns. Intuitively, before searching the supporting rows of an order, we first check whether all its sub-orders are supported by at least r_{min} rows. If not, we then confirm that the order does not give rise to any valid POPSM and, thus, can be discarded. This idea motivates us to develop an Apriori-based framework to mine POPSM patterns, as detailed in Algorithm 1.

In Algorithm 1, if a size-k order has at least r_{min} supporting rows, we call it a *size-k frequent order*. Any size-k order can be a *size-k candidate order*. We denote by $\mathcal{C}_k$ and $\mathcal{F}_k$ a set of size-k candidate orders and the set of all size-k frequent orders, respectively. First, the set of size-1 frequent orders, i.e., $\mathcal{F}_1$, is generated, which contains orders with every single column in T. Then, the GENCAND procedure detailed in Section 5.2 is invoked to generate $\mathcal{C}_k$ from $\mathcal{F}_{k-1}$. The candidate generation algorithm GENCAND aims to reduce the size of $\mathcal{C}_k$ as much as possible, but still guarantees that all the size-k

$\mathcal{F}_3$: five size-3 frequent orders

τ_1	$[t_1 \prec t_2 \prec t_3]$
τ_2	$[t_1 \prec t_2 \prec t_4]$
τ_3	$[t_1 \prec t_3 \prec t_4]$
τ_4	$[t_2 \prec t_3 \prec t_4]$
τ_5	$[t_2 \prec t_4 \prec t_3]$

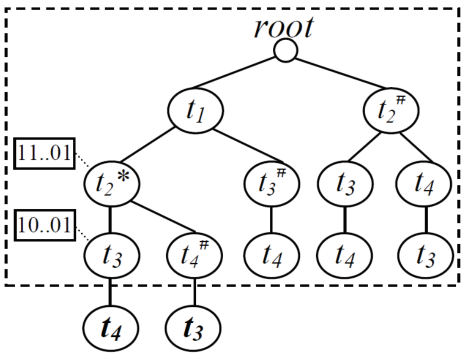


Fig. 4. A *CandTree* example

frequent orders are included in $\mathcal{C}_k$. Next, the COUNTSUP procedure detailed in Section 5.3 is invoked to verify the candidate orders in $\mathcal{C}_k$. Specifically, the COUNTSUP procedure counts the number of supporting rows for each candidate order. Those candidate orders that are supported by at least r_{min} rows are called the size-k frequent orders and form the set $\mathcal{F}_k$. If k is larger than or equal to c_{min} and $\mathcal{F}_k$ is not empty, the POPSM patterns $(P, Q : \tau_Q)$ are placed in an output pool, where τ_Q is a frequent order in $\mathcal{F}_k$, and P is the set of supporting rows of τ_Q. Finally, we output those maximal POPSM patterns as the result.

5.2. Candidate Generation

The GENCAND procedure generates the set of size-k candidate orders $\mathcal{C}_k$. This procedure applies to both the UniDist matrix and the NormDist matrix.

Since the candidate order set $\mathcal{C}_k$ must contain all the size-k frequent orders, we may simply include in $\mathcal{C}_k$ all possible size-k orders. However, the size of $\mathcal{C}_k$ increases exponentially when k gets larger. As all the candidate orders in $\mathcal{C}_k$ need to be further verified in COUNTSUP which is a time-consuming process, we expect to exclude from $\mathcal{C}_k$ as many as possible orders which cannot be frequent. The anti-monotonic property is utilized to achieve this goal. Based on the property, a size-k order cannot be a frequent order if any of its size-$(k-1)$ sub-order is not frequent. Therefore, the search space for size-k candidate orders can be restricted to the size-k orders, of which all the size-$(k-1)$ sub-orders are in $\mathcal{F}_{k-1}$. The GENCAND procedure is thus designed to generate $\mathcal{C}_k$ based on $\mathcal{F}_{k-1}$.

A prefix-sharing tree structure called *CandTree* is employed to organize frequent or candidate orders, and the candidate generation process is implemented by updating the tree. Figure 4 presents a *CandTree* example, where five size-3 frequent orders are listed in the left table, and the subtree inside the rectangle in the right diagram is the corresponding *CandTree*. In the *CandTree*, all the orders are organized in the prefix-sharing manner, and a path leading from *root* to a leaf node exactly corresponds to an order. For example, the rightmost path leading from *root* to nodes t_2, t_4, and finally t_3 corresponds to the order τ_5 in the table. Each leaf node in the *CandTree* is associated with a bitmap, where the ith bit corresponds to the ith row in the input probabilistic matrix. It is set to 1 if the probabilistic support of row g_i to the corresponding order is larger than or equal to the threshold α; otherwise, it is set to 0. Note that the nonleaf nodes in the *CandTree* of $\mathcal{F}_{k-1}$ were once leaf nodes in the *CandTree* of $\mathcal{F}_r$ with $r < (k-1)$. Thus, every node in the *CandTree* except *root* is actually associated with a bitmap.

ALGORITHM 2: GENCAND

```
Input: CandTree($\mathcal{F}_{k-1}$)
Output: CandTree($\mathcal{C}_k$)
Variable: $\tau[p]$ - the order corresponding to the path from root to $p$
while there are unvisited nodes in CandTree($\mathcal{F}_{k-1}$) do
    Depth-first traversal to node $p$ with $|\tau[p]| = k - 2$;
    for $p$'s child nodes $t_1$ and $t_2$ do
        Insert $t_2$ as a child of $t_1$;
    end
end
for all newly inserted leaf nodes $t$ do
    if any size-$(k-1)$ sub-order of $\tau[t] \notin \mathcal{F}_{k-1}$ then
        Undo insertion of node $t$;
    end
end
Prune the subtrees not containing any newly-inserted leaf node.
```

The GENCAND procedure takes the *CandTree* of $\mathcal{F}_{k-1}$ as input and returns an updated *CandTree*, where each path leading from *root* to a leaf node exactly corresponds to a size-k candidate order. We call the updated tree the *CandTree* of $\mathcal{C}_k$. GENCAND consists of three steps, and the details are shown in Algorithm 2. First, given the *CandTree* of $\mathcal{F}_{k-1}$, a depth-first traversal is carried out. When there are still unvisited nodes, the tree traversal continues until a non-leaf node p is reached such that the number of nodes along the path leading from *root* to p (with *root* excluded) is $k-2$. For each pair of p's child nodes t_1 and t_2, t_2 is inserted as a child of t_1. In this step, instead of inserting all the size-k orders, we only insert those orders, of which the two size-$(k-1)$ sub-orders that share a size-$(k-2)$ prefix are frequent. The search space of size-k candidate orders is thus reduced. After finishing the tree traversal, for each newly inserted node t, the path leading from *root* to t forms a size-k order $\tau[t]$. Then, a pruning step is carried out to further examine whether all the size-$(k-1)$ sub-orders of $\tau[t]$ are in $\mathcal{F}_{k-1}$. If not, $\tau[t]$ is surely not a frequent order, and can be excluded from $\mathcal{C}_k$ as well. Thus, the insertion of node t is revoked. Since the size-$(k-1)$ frequent orders are stored in the *CandTree* of $\mathcal{F}_{k-1}$, the pruning step can be efficiently accomplished by scanning the *CandTree*. After pruning, the remaining size-k orders form the set of size-k candidate orders $\mathcal{C}_k$. Finally, the tree is trimmed by deleting those subtrees that do not contain any newly inserted leaf node, since these subtrees do not lead to any valid size-k candidate orders. The updated tree is the *CandTree* of $\mathcal{C}_k$.

We further illustrate GENCAND using the example in Figure 4. The *CandTree* of $\mathcal{F}_3$ is traversed until node t_2, which is marked with "$*$", is reached. For the two child nodes of t_2 (i.e., t_3 and t_4), one is inserted as a child of the other. Then, the tree traversal continues until all the nodes are visited. Finally, only two new leaf nodes (i.e. t_3 and t_4) are inserted, which are highlighted in bold font. For the newly inserted node t_4, the path leading from *root* to it represents the order $\tau[t_4] = [t_1 \prec t_2 \prec t_3 \prec t_4]$. Since all the size-3 sub-orders of $\tau[t_4]$, i.e., τ_1, τ_2, τ_3 and τ_4 in the table of $\mathcal{F}_3$, are frequent, $\tau[t_4]$ is a size-4 candidate order. Similarly, the path leading from *root* to the other newly inserted node t_3 corresponds to another size-4 order $\tau[t_3] = [t_1 \prec t_2 \prec t_4 \prec t_3]$. One of the size-3 sub-orders of $\tau[t_3]$, i.e., $[t_1 \prec t_4 \prec t_3]$, is not in $\mathcal{F}_3$, which means that it is not frequent. Thus, the order $\tau[t_3]$ is surely not a frequent order and the insertion of node t_3 is revoked. Finally, we prune three subtrees with their root nodes marked with "#", since none of these three subtrees contains any newly inserted leaf node and so they would not lead to any size-4 candidate order. The updated tree is the *CandTree* of $\mathcal{C}_4$.

Notably, GENCAND is said to be *complete*, in the sense that all size-k frequent orders are included in $\mathcal{C}_k$. We now justify its completeness.

Assume that $\mathcal{F}_{k-1}$ contains all size-$(k-1)$ frequent orders. Let $\tau = [t_1 \prec \cdots \prec t_k]$ be a size-k frequent order. According to the anti-monotonic property of the probabilistic support, all the supporting rows of τ should also be the supporting rows of the size-$(k-1)$ sub-orders of τ. In other words, if τ is a frequent order, all the size-$(k-1)$ sub-orders of τ must also be frequent and in $\mathcal{F}_{k-1}$. Suppose there are $\tau_1 = [t_1 \prec \cdots \prec t_{k-2} \prec t_{k-1}]$ and $\tau_2 = [t_1 \prec \cdots \prec t_{k-2} \prec t_k]$, which are two frequent size-$(k-1)$ sub-orders of τ. Since τ_1 and τ_2 only differ in the last item, the node that corresponds to item t_{k-1} in τ_1 and the node that corresponds to item t_k in τ_2 are two leaf nodes in the *CandTree* of $\mathcal{F}_{k-1}$, and they share the same parent node t_{k-2}. According to GENCAND, a new node t_k is then inserted as a child node of t_{k-1}, and the path leading from *root* to the newly inserted t_k corresponds to the order τ. As all the size-$(k-1)$ sub-orders of τ are frequent and in $\mathcal{F}_{k-1}$, the insertion of the new node t_k does not get revoked in the pruning step. Thus, the order τ is generated and added to $\mathcal{C}_k$.

5.3. Support Counting

After the set of size-k candidate orders $\mathcal{C}_k$ is generated, the COUNTSUP procedure is invoked to verify whether the orders in $\mathcal{C}_k$ are frequent or not. Dynamic programming (DP) techniques are developed to compute the probabilistic support of a row w.r.t. a candidate order. If the probabilistic support of at least r_{min} rows w.r.t. a candidate order is larger than or equal to the support threshold α, the candidate order is called frequent. The frequent candidate orders form the size-k frequent order set $\mathcal{F}_k$.

We develop two DP techniques, respectively called *static DP* (SDP) and *dynamic DP* (DDP), for computing the probabilistic support of a row w.r.t. a candidate order. SDP disregards the size of the candidate order to which the probabilistic support is computed, and breaks down the computation into a fixed number of subproblems. In contrast, for a candidate order with smaller size, DDP breaks down the computation of its probabilistic support into a fewer number of subproblems. Furthermore, when these two techniques are adopted to compute the probabilistic support w.r.t. the same candidate order, the number of subproblems divided by DDP is no more than the number of subproblems divided by SDP. Thus, compared to SDP, DDP saves computational cost by solving fewer number of subproblems. However, DDP has to spend some extra time for dynamically forming the subproblems when running the process of dynamic programming. Our experiments in Section 6.4 show that both techniques exhibit their efficiency under different scenarios. For the sake of clarity, when we employ SDP or DDP to compute the probabilistic support of a row w.r.t. a candidate order in a UniDist matrix, we respectively call them the UNISDP and UNIDDP methods. When they are employed to compute the probabilistic support in a NormDist matrix, we respectively call them the NORMSDP and NORMDDP methods. Table III summarizes the four DP methods corresponding to different probabilistic matrices.

Table III. The DP methods corresponding to different types of probabilistic matrices

DP Technique \ Matrix Type	UniDist Matrix	NormDist Matrix
Static DP	UNISDP	NORMSDP
Dynamic DP	UNIDDP	NORMDDP

For either SDP or DDP, its recursive DP equations for the two types of matrices are similar, except that more complex preprocessing steps are needed when it is applied to

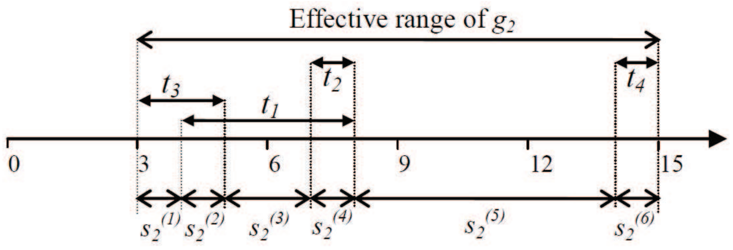


Fig. 5. The effective range $[y_2, x_2] = [3, 15]$ of row g_2 (in Table I) and its six subranges $\{s_2^{(1)}, \ldots, s_2^{(6)}\}$

the NormDist matrix. For simplicity in presentation, the formulation of the DP equations which is common for UniSDP and NormSDP (or respectively for UniDDP and NormDDP) is detailed when we introduce UniSDP (or respectively UniDDP). When introducing NormSDP and NormDDP, we just focus on explaining the different preprocessing steps adopted by them. Then, we show that the COUNTSUP procedure combines any of the DP methods with the traversal of *CandTree* to compute the probabilistic support of a row w.r.t. all the candidate orders in $\mathcal{C}_k$ in an efficient way.

5.3.1. Static DP for UniDist matrix - **UNISDP**. We now detail the UNISDP method for computing the probabilistic support of a row w.r.t. an order in an m-by-n UniDist matrix $M^U(G,T)$. Before applying the dynamic programming technique, there are two preprocessing steps to be carried out as follows:

(1) First, for the record of each row g_i, i.e., $\{[l_{i1}, u_{i1}], \ldots, [l_{in}, u_{in}]\}$, we define $x_i = \max\{u_{ij} | 1 \leq j \leq n\}$ and $y_i = \min\{l_{ij} | 1 \leq j \leq n\}$. The range $[y_i, x_i]$ is called the *effective range* of g_i. Every range entry $[l_{ij}, u_{ij}]$ with $1 \leq j \leq n$ is fully covered by the effective range, and the probability that any corresponding random variable M_{ij} is located outside the effective range is 0.
(2) Then, all l_{ij}'s and u_{ij}'s with $1 \leq j \leq n$ are sorted in non-decreasing order, and these $2n$ values divide the range $[y_i, x_i]$ into p_i subranges $s_i^{(j)}$ with $1 \leq j \leq p_i$. The number of subranges p_i is at most $2n-1$. We sort the p_i subranges in nondecreasing order of their middle point values, and get an ordered subrange set $\mathcal{S}_i = \langle s_i^{(1)}, \ldots, s_i^{(p_i)} \rangle$ corresponding to row g_i. Note that for any two adjacent subranges in $\mathcal{S}_i$, there exists at least one random variable in $\{M_{i1}, \ldots, M_{in}\}$ such that, its probability densities in these two subranges are different.

Figure 5 shows the effective range of row g_2 in Table I which is $[3, 15]$. The effective range is divided into six subranges that are sorted and indicated as $\langle s_2^{(1)}$, ..., $s_2^{(6)} \rangle$.

Given a row g_i with its ordered subrange set $\mathcal{S}_i = \langle s_i^{(1)}, \ldots, s_i^{(p_i)} \rangle$, and a size-$k$ order $\tau_Q = [t_{j_1} \prec t_{j_2} \prec \cdots \prec t_{j_k}]$, the probabilistic support of g_i w.r.t. τ_Q, i.e., $\mathcal{PS}(g_i, \tau_Q) = P(M_{ij_1} < \cdots < M_{ij_k})$, is computed as follows.

We first create a table $Z^U(R, C)$ with k rows and p_i columns for dynamic programming. The xth row r_x corresponds to the size-x prefix sub-order of τ_Q, i.e., $[t_{j_1} \prec \cdots \prec t_{j_x}]$, and the yth column c_y corresponds to the first y subranges in $\mathcal{S}_i$, i.e., $\langle s_i^{(1)}, \ldots, s_i^{(y)} \rangle$. The entry $Z^U(r_x, c_y)$ is the probability of the random event $(M_{ij_1} < \cdots < M_{ij_x})$ with $M_{ij_1}, \ldots, M_{ij_x}$ located in the first y subranges. Thus, the entry $Z^U(r_k, c_{p_i})$ is the probability of the event $(M_{ij_1} < \cdots < M_{ij_k})$ with $M_{ij_1}, \ldots, M_{ij_k}$ in all the p_i subranges, which actually is $\mathcal{PS}(g_i, \tau_Q)$. Further, the entry $Z^U(r_x, c_y)$ equals the sum of the following three parts:

(1) the probability of $(M_{ij_1} < \cdots < M_{ij_x})$ with $M_{ij_1}, \ldots, M_{ij_x}$ in the first $(y-1)$ subranges; and
(2) the probability of $(M_{ij_1} < \cdots < M_{ij_x})$ with $M_{ij_1}, \ldots, M_{ij_x}$ in the yth subrange; and
(3) the probability of $(M_{ij_1} < \cdots < M_{ij_z})$ with $1 \leq z < x$ and $M_{ij_1}, \ldots, M_{ij_z}$ in the first $(y-1)$ subranges, multiplied by the probability of $(M_{ij_{(z+1)}} < \cdots < M_{ij_x})$ with $M_{ij_{(z+1)}}, \ldots, M_{ij_x}$ in the yth subrange $s_i^{(y)}$.

The dynamic programming equation can then be written as:

$$Z^U(r_x, c_y) = Z^U(r_x, c_{y-1}) + P(M_{ij_1} < \cdots < M_{ij_x} | s_i^{(y)}) + \sum_{z=1}^{x-1} Z^U(r_z, c_{y-1}) P(M_{ij_{(z+1)}} < \cdots < M_{ij_x} | s_i^{(y)}), \tag{5}$$

where $P(M_{ij_1} < \cdots < M_{ij_x} | s_i^{(y)})$ is the probability of the event $(M_{ij_1} < \cdots < M_{ij_x})$ with $M_{ij_1}, \ldots, M_{ij_x}$ in the subrange $s_i^{(y)}$. As $M_{ij_1}, \ldots, M_{ij_x}$ all follow uniform distributions with the PDFs $p_{ij_1}, \ldots, p_{ij_x}$, $P(M_{ij_1} < \cdots < M_{ij_x} | s_i^{(y)})$ is then computed as

$$P(M_{ij_1} < \cdots < M_{ij_x} | s_i^{(y)}) = \frac{|s_i^{(y)}|^x}{x!} \cdot \prod_{k=1}^{x} \left[p_{ij_k} I(s_i^{(y)} \subseteq [l_{ij_k}, u_{ij_k}]) \right],$$

where $I(\cdot)$ is the indicator function in usual convention.

Assume that the maximal length of the mined patterns is L, which is usually much smaller than the number of columns n in the UniDist matrix. The time complexity for constructing the DP table $Z^U(R, C)$, i.e., the time for computing the probabilistic support of a row w.r.t. an order, is $O(L^2 \times (2n-1)) = O(L^2 n)$. However, we will show in Section 5.3.5 that, by combining the computation of the DP table with the traversal of the *CandTree*, the time for computing the probabilistic support of a row w.r.t. an order is only $O(Ln)$.

5.3.2. Dynamic DP for UniDist Matrix - **UNIDDP**. In the UNISDP method, each row g_i is converted to at most $(2n-1)$ subranges based on the range entries of g_i under T. Accordingly, the table constructed for dynamic programming contains at most $(2n-1)$ columns, which makes the DP process time-consuming.

However, we find that the associated column set Q of a candidate order τ_Q is usually a small subset of T, and, with $x_i^Q = \max\{l_{ij}, u_{ij} | t_j \in Q\}$ and $y_i^Q = \min\{l_{ij}, u_{ij} | t_j \in Q\}$, the probability that the random variable M_{ij} with $t_j \in Q$ falls out of the range $[y_i^Q, x_i^Q]$ is 0. Thus, we call $[y_i^Q, x_i^Q]$ the effective range of g_i over Q, or simply the *Q-based effective range*. Since $[y_i^Q, x_i^Q]$ is a subrange of $[y_i, x_i]$, instead of applying the DP technique over $[y_i, x_i]$, we only need to apply it over the smaller range $[y_i^Q, x_i^Q]$ to obtain the probabilistic support of g_i w.r.t. τ_Q.

In addition, we find that there may exist pairs of adjacent subranges in the ordered subrange set $\mathcal{S}_i$, such that the probability density of M_{ij} (with $t_j \in Q$) in the adjacent subranges are the same. During the dynamic programming process, such adjacent subranges can be merged, which accordingly leads to a smaller DP table.

Let us consider row g_2 shown in Figure 5 again. Suppose that the probabilistic support of g_2 w.r.t. an order $\tau_Q = [t_3 \prec t_1]$ with $Q = \{t_1, t_3\}$ is computed. We get the Q-based effective range $[y_2^Q, x_2^Q]$, which is $[3, 8]$, and know that the probability densities of the random variables M_{23} and M_{21} outside $[y_2^Q, x_2^Q]$ are both 0. Furthermore, for the two adjacent subranges $s_2^{(3)}$ and $s_2^{(4)}$, the probability density of M_{23} in them are

ALGORITHM 3: UniDDP

```
Input: Row $g_i$ with the record $\{[l_{i1}, u_{i1}], \ldots, [l_{in}, u_{in}]\}$; a size-$k$ order $\tau_Q = [t_{j_1} \prec t_{j_2} \prec \cdots \prec t_{j_k}]$
Output: The probabilistic support of $g_i$ w.r.t. $\tau_Q$
Variable: The ordered subrange set $\mathcal{S}_i = \langle s_i^{(1)}, \ldots, s_i^{(|\mathcal{S}_i|)} \rangle$
$Q_{:1} = \{t_{j_1}\}$, and $\tau_{Q:1} = [t_{j_1}]$;           // $\tau_{Q:1}$ is the size-1 prefix sub-order of $\tau_Q$
Initialize $\mathcal{S}_i$ to $\langle [l_{ij_1}, u_{ij_1}] \rangle$;
Construct a DP table $Z^U(R, C)$ with $|Q_{:1}|$ rows and $|\mathcal{S}_i|$ columns;
for $x = 2$ to $k$ do
    $Q_{:x} = Q_{:(x-1)} \cup t_{j_x}$ and $\tau_{Q:x} = [t_{j_1} \prec \cdots \prec t_{j_x}]$;
    Insert a row in $Z^U$ after the last row ;           // corresponds to newly appended $t_{j_x}$
    for $v \in \{l_{ij_x}, u_{ij_x}\}$ do
        if there exists $s_i^{(y)} \in \mathcal{S}_i$ such that $v \in s_i^{(y)}$ then
            Update $\mathcal{S}_i$ by splitting $s_i^{(y)}$ into two subranges $s_i^{(y_1)}$ and $s_i^{(y_2)}$;
            Insert a column in $Z^U$ before the $y$th column;
        else if $v <$ the lower bound of $s_i^{(1)}$ then
            Insert a new subrange $[v$, lower bound of $s_i^{(1)}]$ in $\mathcal{S}_i$ before $s_i^{(1)}$;
            Insert a column in $Z^U$ before the first column;
        else if $v >$ the upper bound of $s_i^{(|\mathcal{S}_i|)}$ then
            Insert a new subrange [upper bound of $s_i^{(|\mathcal{S}_i|)}, v]$ in $\mathcal{S}_i$ after $s_i^{(|\mathcal{S}_i|)}$;
            Insert a column in $Z^U$ after the last column ;
        end
    end
    Compute the inserted $(2|Q_{:x}| + |\mathcal{S}_i| - 2)$ entries in $Z^U$ using Formula (5);
end
Return $Z^U(r_k, c_{|\mathcal{S}_i|})$;           // stores the probabilistic support of $g_i$ to $\tau_Q$
```

both 0 and the probability density of M_{21} in them are both 0.25. Therefore, these two adjacent subranges can be merged into one, and the Q-based effective range $[y_2^Q, x_2^Q]$ is accordingly divided into the three subranges of $[3, 4]$, $[4, 5]$, and $[5, 8]$.

Motivated by the above observation, we propose to dynamically form subranges during the DP process. This idea leads to the development of a dynamic DP method called UniDDP. Generally, in order to compute the probabilistic support of a row g_i w.r.t. a size-k order τ_Q, UniDDP computes the probabilistic support of g_i w.r.t. the size-x prefix sub-order of τ_Q by utilizing the DP table constructed for the size-$(x-1)$ prefix sub-order until x increases from 1 to k.

The details of the UniDDP method are shown in Algorithm 3. The record of row g_i, i.e., $\{[l_{i1}, u_{i1}], \ldots, [l_{in}, u_{in}]\}$, and a size-$k$ order $\tau_Q = [t_{j_1} \prec t_{j_2} \prec \cdots \prec t_{j_k}]$ are taken as an input. UniDDP starts by computing the probabilistic support of g_i w.r.t. the size-1 prefix sub-order of τ_Q, which we denote by $\tau_{Q:1} = [t_{j_1}]$ with $Q_{:1} = \{t_{j_1}\}$. The ordered subrange set $\mathcal{S}_i$ is initialized with a single subrange $[l_{ij_1}, u_{ij_1}]$. Then, a DP table $Z^U(R, C)$ corresponding to $\tau_{Q:1}$ and containing $|Q_{:1}|$ rows and $|\mathcal{S}_i|$ columns is constructed. The DP mechanism of UniDDP is the same as that of UniSDP. That is, the xth row in Z^U corresponds to the size-x prefix sub-order of τ_Q, i.e., $\tau_{Q:x}$, the yth column corresponds to the first y subranges in $\mathcal{S}_i$, and the entry $Z^U(r_x, c_y)$ is the probability of the random event $(M_{ij_1} < \cdots < M_{ij_x})$ with $M_{ij_1}, \ldots, M_{ij_x}$ in the first y subranges. Suppose we denote by Z^U_{x-1} the DP table corresponding to $\tau_{Q:(x-1)}$. UniDDP efficiently gets the DP table Z^U_x based on Z^U_{x-1} in the following way. First, a new row is inserted in Z^U_{x-1} after the last row, which corresponds to the xth item t_{j_x} in $\tau_{Q:x}$.

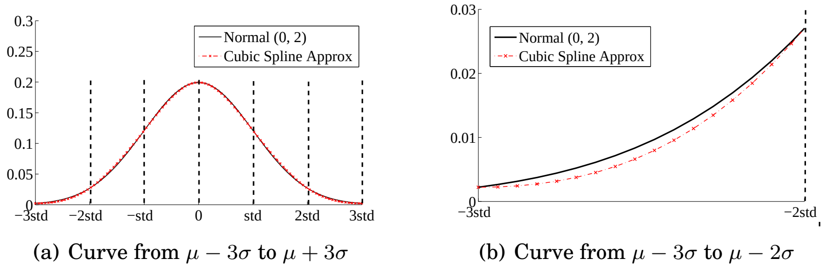


Fig. 6. Approximation of the normal distribution $(0,2)$ using a cubic spline with 6 pieces

Then, the range $[l_{ij_x}, u_{ij_x}]$ is imposed on the $Q_{:(x-1)}$-based effective range, which is already divided into a set of subranges and stored in $\mathcal{S}_i$. If l_{ij_x} or u_{ij_x} falls within some existing subrange, say $s_i^{(y)}$, in $\mathcal{S}_i$, it splits $s_i^{(y)}$ into two new subranges $s_i^{(y_1)}$ and $s_i^{(y_2)}$. $\mathcal{S}_i$ is then updated with $s_i^{(y)}$ being replaced by $s_i^{(y_1)}$ and $s_i^{(y_2)}$. As these two new subranges are ranked y and $(y+1)$ in $\mathcal{S}_i$, a new column is inserted in the DP table before the yth column. The new yth column corresponds to the first y subranges in the updated $\mathcal{S}_i$, i.e., $\langle s_i^{(1)}, \ldots, s_i^{(y-1)}, s_i^{(y_1)}\rangle$. The new $(y+1)$th column corresponds to the first $y+1$ subranges in the updated $\mathcal{S}_i$ which actually cover the same range as the first y subranges in $\mathcal{S}_i$ before splitting. Thus, the columns after the yth column in the updated DP table do not need to be recomputed. If l_{ij_x} or u_{ij_x} falls before $s_i^{(1)}$ or after $s_i^{(|\mathcal{S}_i|)}$, a new subrange is inserted to $\mathcal{S}_i$ before $s_i^{(1)}$ or after $s_i^{(|\mathcal{S}_i|)}$, and a new column corresponding to this new subrange is inserted in Z_x^U. Similarly, except the newly inserted column, none of the other columns in the updated DP table needs to be recomputed. Finally, Formula (5) is employed to compute the values of newly inserted entries in the DP table Z^U, and the updated Z^U is the DP table corresponding to the size-x prefix sub-order. When x equals k, the entry $Z^U(r_k, c_{|\mathcal{S}_i|})$ stores the probabilistic support of row g_i w.r.t. the size-k order τ_Q.

The DP table constructed by UNIDDP contains at most $(2L-1)$ columns with L as the maximal length of the mined patterns. Thus, the time cost incurred by UNIDDP is $O(L^2 \times (2L-1)) = O(L^3)$. Similarly, by combining the computation of the DP table with the traversal of the *CandTree*, the time cost for computing the probabilistic support of a row w.r.t. an order is reduced to $O(L^2)$.

5.3.3. Static DP for NormDist Matrix - **NORMSDP***.* While it is computationally inefficient to directly deal with the normal distribution when computing the probabilistic support, we propose to adopt the spline technique to approximate the normal distribution as piecewise low-degree polynomials. This approach is desirable, since spline interpolation guarantees good approximation ratio, and the technique has been well studied in the area of approximation theory [Heath 2002]. Figure 6(a) shows a cubic spline approximation of a normal distribution having the mean value $\mu = 0$ and the standard deviation $\sigma = 2$. The approximation range is $[\mu - 3\sigma, \mu + 3\sigma]$, and it is uniformly divided into six intervals. Within each interval, the density function is expressed using a third-order polynomial. For example, the polynomial corresponding to the interval $[-3\sigma, -2\sigma]$ is $f(x) = 0.001x^3 - 0.004x^2 + 0.002$. Figure 6(b) focuses on the part of the curve on the interval $[\mu - 3\sigma, \mu - 2\sigma]$.

Note that, although the domain of a normal distribution spans the whole real line, the approximation range with 2σ from the mean already accounts for about 95% of the distribution. For many types of uncertain data such as noisy gene expression data, this approximation range should be able to cover the majority of the true replicates in practice. Thus, we simply assume that the probability density outside the approximation range is 0.

Next, we introduce the NORMSDP method which takes the SDP technique to compute the probabilistic support of a row w.r.t. a candidate order in the NormDist matrix. The preprocessing consists of the following three steps:

(1) First, the cubic spline technique is used to convert every entry $M^N_{ij} = (\mu_{ij}, \sigma_{ij})$ in a NormDist matrix to a set of polynomials. Specifically, we fix the approximation range to $[\mu_{ij} - t * \sigma_{ij}, \mu_{ij} + t * \sigma_{ij}]$ where t can be set to 1, 2, or 3. To guarantee the approximation ratio as well as the computational efficiency, the number of intervals q for cubic spline approximation is commonly set to 4 or 6 in practice. Then, the approximation range of every normal distribution is divided into q intervals by $q + 1$ breakpoints, and each interval is approximated by a third-order polynomial. Any entry M^N_{ij} in the NormDist matrix can thus be represented as (B_{ij}, F_{ij}), where $B_{ij} = \{b^{(1)}_{ij}, \ldots, b^{(q+1)}_{ij}\}$ is the set of $q + 1$ breakpoints and $F_{ij} = \{f^{(1)}_{ij}(\cdot), \ldots, f^{(q)}_{ij}(\cdot)\}$ is the set of q polynomials corresponding to each interval.
(2) Second, for every row g_i with its record $\{(B_{i1}, F_{i1}), \ldots, (B_{in}, F_{in})\}$, we define $x_i = \max\{b^{(k)}_{ij} | 1 \leq k \leq q + 1, 1 \leq j \leq n\}$ and $y_i = \min\{b^{(k)}_{ij} | 1 \leq k \leq q + 1, 1 \leq j \leq n\}$, and call $[y_i, x_i]$ the effective range of g_i over T. Then, all the breakpoints $b^{(k)}_{ij}$ with $1 \leq k \leq q + 1$ and $1 \leq j \leq n$ are sorted in non-decreasing order, and these $n(q + 1)$ breakpoints divide $[y_i, x_i]$ into p_i subranges $s^{(k)}_i$ with $1 \leq k \leq p_i$. The number of subranges p_i is at most $n(q + 1) - 1$.
(3) Finally, similarly to the second preprocessing step of UNISDP, we get the ordered subrange set of row g_i, and denote it by $\mathcal{S}_i = \langle s^{(1)}_i, \ldots, s^{(p_i)}_i \rangle$. For an entry M^N_{ij}, its probability density within any subrange $s^{(k)}_i$ is 0 if $s^{(k)}_i$ is disjoint from its approximation range, or is $f^{(r)}_{ij}(\cdot)$ if $s^{(k)}_i \subseteq [b^{(r)}_{ij}, b^{(r+1)}_{ij}]$.

After preprocessing, the DP process of NORMSDP is similar to that of UNISDP. Specifically, in order to compute the probabilistic support of a row g_i w.r.t. a size-k order τ_Q, a k-by-p_i DP table $Z^N(R, C)$ is first created, with the xth row corresponding to the size-x prefix sub-order of τ_Q and the yth column corresponding to the first y subranges in $\mathcal{S}_i$. Then, Formula (5) is employed to incrementally compute the support. The only difference is that, in the subranges, each random variable is now associated with a polynomial (or 0) in NORMSDP, while it is associated with a constant (or 0) in UNISDP.

As the number of intervals q is set to 4 or 6 during cubic spline approximation, at most $5n$ or $7n$ subranges are generated for each row in the third preprocessing step. In addition, the time for computing an entry in the Z^N table is $O(L^2)$. Thus, the time cost for computing the Z^N table in NORMSDP is $O(L^3 n)$.

5.3.4. Dynamic DP for NormDist Matrix - **NORMDDP**. As the DP table constructed by NORMSDP contains up to $(q + 1)n$ columns, which makes the DP process quite time-consuming, we similarly adopt the DDP technique to reduce the size of the DP table.

Generally, after preprocessing, the NORMDDP method also adopts an incremental way to compute the probabilistic support of a row g_i w.r.t. a size-k order τ_Q. It constructs the DP table corresponding to the size-x prefix sub-order of τ_Q, denoted as $\tau_{Q_{:x}}$, based on the DP table corresponding to the order $\tau_{Q_{:(x-1)}}$. When x increases to k, the entry in the last row and last column of the DP table stores the probabilistic support of g_i

w.r.t. τ_Q. Since the NORMDDP method is similar to UNIDDP, we omit the details of NORMDDP.

We denote by Z^N_{x-1} the DP table corresponding to the order $\tau_{Q_{:(x-1)}}$. The $Q_{:(x-1)}$-based effective range of g_i is acquired by setting the lower and upper bounds respectively to $\min\{b^{(k)}_{ij} | 1 \leq k \leq q+1, t_j \in Q_{:(x-1)}\}$ and $\max\{b^{(k)}_{ij} | 1 \leq k \leq q+1, t_j \leq Q_{:(x-1)}\}$. The $Q_{:(x-1)}$-based effective range is divided into subranges that are sorted and stored in $\mathcal{S}_i = \langle s^{(1)}_i, s^{(2)}_i, \ldots \rangle$. The DP table for the order $\tau_{Q_{:x}}$ is then constructed in the following way. First, a new row is inserted in Z^N_{x-1} after the last row, which corresponds to the xth item t_{j_x} in $\tau_{Q_{:x}}$. Then, the approximation range of the entry M_{ij_x}, which is interpolated and represented as (B_{ij_x}, F_{ij_x}), is imposed on the $Q_{:(x-1)}$-based effective range. Similarly to the UNIDDP method, for each breakpoint $b^{(z)}_{ij_x}$ in B_{ij_x}, NORMDDP checks whether it splits some existing subrange in $\mathcal{S}_i$ or introduces a new subrange to $\mathcal{S}_i$. Every update in $\mathcal{S}_i$ is followed by an insertion of a new column in the DP table. As there are $(q+1)$ breakpoints in B_{ij_x}, at most $(q+1)$ new columns will be inserted in the DP table. Finally, Formula (5) is adopted to compute the values of newly inserted entries, and the resultant updated DP table is Z^N_x that corresponds to the order $\tau_{Q_{:x}}$.

As the DP table constructed by NORMDDP contains at most $((q+1)L-1)$ columns, with L as the maximum length of candidate orders, the time cost by NORMDDP is $O(L^4)$.

5.3.5. **COUNTSUP**. The COUNTSUP procedure combines any version of the four DP methods introduced above with the traversal of the *CandTree*, and efficiently computes the probabilistic support of a row w.r.t. all the candidate orders. The details of COUNTSUP are given in Algorithm 4.

The input matrix $M(G,T)$ is either a UniDist matrix or a NormDist matrix. Accordingly, the DP table is either $Z^U(R,C)$ or $Z^N(R,C)$. For each row g_i in G, the *CandTree* is traversed in the depth-first manner. When a non-leaf node p is reached and if the ith bit in the bitmap of p is 1, it means that the probabilistic support of row g_i w.r.t. the order $\tau[p]$ satisfies the threshold. An SDP or DDP method is then used to compute the support. The SDP methods, such as UNISDP and NORMSDP, employ the DP function shown in Formula (5) to compute the entries in the lth row of the DP table, where l is the number of items in the order $\tau[p]$. The DDP methods, such as UNIDDP and NORMDDP, first update the current DP table to make it correspond to the order $\tau[p]$, and then employ the DP function to compute the values of inserted entries. After the probabilistic support of g_i w.r.t. $\tau[p]$ is computed, the COUNTSUP method continues to traverse the children of node p. On the other hand, if the ith bit in the bitmap of p is 0, it means that the probabilistic support of g_i w.r.t. $\tau[p]$ does not satisfy the threshold. Due to the anti-monotonic property, the probabilistic supports of g_i w.r.t. all the orders with $\tau[p]$ as a prefix would not satisfy the threshold either, and thus COUNTSUP stops traversing the subtree rooted at p and backtracks to the next unvisited node. If either DDP method is adopted, the update operations over the DP table for the nodes along the backtracking route need to be revoked. If a leaf node p is reached, one of the DP methods is adopted to compute the DP table, and the probabilistic support of row g_i w.r.t. the candidate order $\tau[p]$, i.e., $\mathcal{PS}(g_i, \tau[p])$, equals $Z(r_l, c_{|\mathcal{S}_i|})$. If $\mathcal{PS}(g_i, \tau[p]) \geq \alpha$, the ith bit in the bitmap of p is set to 1; otherwise, it is set to 0. The size-k frequent order set $\mathcal{F}_k$ contains all the orders corresponding to the path from *root* to the leaf nodes, of which the number of 1's in the bitmap is at least r_{min}.

When UNISDP or UNIDDP is adopted in COUNTSUP, we call the PROBAPRI algorithm UNIAPRI; when NORMSDP or NORMDDP is adopted, we call the PROBAPRI algorithm NORMAPRI. Recall that the difference between the DP methods for the UniDist matrix and the DP methods for the NormDist matrix is that, the PDFs associ-

ALGORITHM 4: COUNTSUP

```
Input: Probabilistic matrix $M(G,T)$; CandTree($\mathcal{C}_k$); support threshold $\alpha$
Variable: DP table $Z(R,C)$; $\mathcal{S}_i$ - the ordered subrange set of row $g_i$; $\tau[p]$ - the order
          corresponding to the path from $root$ to $p$
for each row $g_i$ in $G$ do
    while there are unvisited nodes in $CandTree(\mathcal{C}_k)$ do
        Traverse to a node $p$ in the depth-first manner;
        if $p$ is a nonleaf node then
            if $p.bm[i] == 1$ then //the bitmap of $p$
                $l = |\tau[p]|$;
                if UNISDP or NORMSDP is adopted then
                    Compute the entries in row $l$ of $Z$;
                else if UNIDDP or NORMDDP is adopted then
                    Update the DP table $Z$ corresponding to $\tau[p]$ ;
                    Compute the values of inserted entries in $Z$;
                end
                Continue to traverse $p$'s children;
            else
                Stop traversing $p$'s children, backtrack to next unvisited node;
                if UNIDDP or NORMDDP is adopted then
                    Revoke the updating of $Z$ for the nodes along the backtracking route;
                end
            end
        else //$p$ is a leaf node
            $l = |\tau[p]|$;
            if UNISDP or NORMSDP is adopted then
                Compute the entries in row $l$ of $Z$;
            else if UNIDDP or NORMDDP is adopted then
                Update the DP table $Z$ corresponding to $\tau[p]$ ;
                Compute the values of inserted entries in $Z$;
            end
            $\mathcal{PS}(g_i,\tau[p]) = Z(r_l, c_{|\mathcal{S}_i|})$;
            if $\mathcal{PS}(g_i,\tau[p]) \geq \alpha$ then
                Set $p.bm[i] = 1$;
            else
                Set $p.bm[i] = 0$;
            end
        end
    end
end
```

ated with the random variables in UNISDP and UNIDDP are constants, while the PDFs associated with the random variables in NORMSDP and NORMDDP are polynomials. Since polynomials are a more general case compared to constants, in this sense, NORMAPRI generalizes UNIAPRI. In addition, the spline technique adopted by NORMAPRI in the preprocessing step is flexible. By using appropriate spline interpolation, NORMAPRI can be adapted to dealing with other probabilistic matrices that have more general continuous distributions.

5.4. Complexity Analysis

We now study the complexity of GENCAND and COUNTSUP, which are the core procedures of the PROBAPRI algorithm.

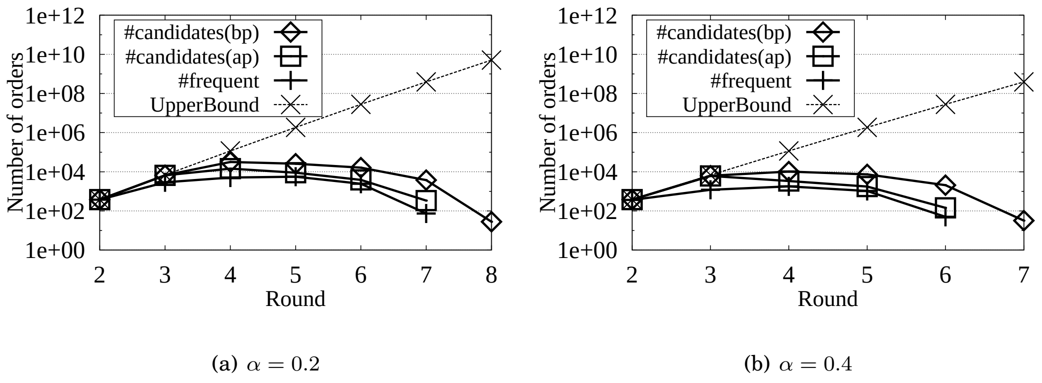


Fig. 7. The effectiveness of GENCAND in controlling the number of candidate orders

The computational cost of the Apriori-based methods is not likely to be well bounded due to the possible exponential number of candidates generated during candidate generation. For example, suppose, in an m-by-n probabilistic matrix, every possible size-c_{min} order of columns is supported by at least r_{min} rows. Then, at the kth round of candidate generation with $1 < k \leq c_{min}$, no size-k order can be pruned by the GENCAND procedure since all its size-$(k-1)$ sub-orders are frequent. The number of size-k candidate orders generated by GENCAND is $\frac{n!}{(n-k)!}$, and it is the upper-bound of the number of candidate orders that can be generated at the kth round. However, such an extreme case rarely happens in real application data, and the Apriori-based methods are usually shown to be practically efficient [Aggarwal et al. 2009; Fang et al. 2012]. Therefore, we conduct an experiment using a real dataset to study the effectiveness of the GENCAND procedure in controlling the number of candidate orders. We use a UniDist matrix with 205 rows and 20 columns. It is converted from a real yeast dataset by adopting the *Std* method with the approximation range set to $[\mu-\sigma, \mu+\sigma]$. The matrix conversion methods are explained in detail in Section 6.1. The POPSM patterns with at least 60 rows and 5 columns are mined from the UniDist matrix with the support threshold α respectively set to 0.2 and 0.4. Given a data matrix with 20 columns, the upper-bound of the number of candidate orders that can be generated at the kth round of candidate generation is $\frac{20!}{(20-k)!}$. Figures 7(a) and 7(b) show the numbers of the candidate orders generated by GENCAND before and after the pruning step, the number of frequent orders, and the corresponding upper-bound at each round under two α settings. The series "#candidates (bp)" and "#candidates (ap)" respectively correspond to the number of candidate orders before and after pruning.

When α equals 0.2, the mining process requires eight rounds as shown in Figure 7(a). At all the rounds, the number of candidate orders after pruning is no more than 4.5 times the number of frequent orders, and it is several orders of magnitude fewer than the corresponding upper-bound starting from the fifth round. At the fourth round, the numbers of candidate orders before and after pruning both reach the maximum, which are respectively 6 times and 3 times the number of frequent orders in the same round. The number of candidate orders before pruning becomes about 50 times the number of frequent orders at the seventh round, which seems to be a big difference. However, the absolute numbers of those candidate orders and frequent orders both decrease substantially in the last several rounds. In addition, the pruning step is efficient, which reduces the number candidate orders to only 4.5 times the number of frequent orders. Similar results can be observed when α equals 0.4. Therefore, GENCAND is efficient in controlling the number of candidate orders during candidate generation.

At the kth round of candidate verification, the COUNTSUP procedure needs to traverse the *CandTree* of $\mathcal{C}_k$ for m times, where m is the number of rows in the input matrix. Since each node in *CandTree* is visited for at most one time in one traversal, at most $km|\mathcal{C}_k|$ nodes are visited where $|\mathcal{C}_k|$ is the size of the set $\mathcal{C}_k$. When a tree node is visited, one of the four DP methods is invoked to update their respective DP tables. We have respectively illustrated the time complexity of the four DP methods in Section 5.3. For ease of presentation, we summarize in Table IV the computational cost of the four DP methods at each node visit, and accordingly compute the cost of COUNTSUP at the kth round. Although the time complexity of COUNTSUP contains up to the fourth power of k which seems time-consuming, k is the size of the candidate orders and thus is usually a small value. Therefore, COUNTSUP is still practically efficient.

Table IV. Time complexity of COUNTSUP corresponding to four DP methods

DP Method	Cost of DP process at a node visit	Cost of COUNTSUP at the kth round
UNISDP	$O(kn)$	$O(k^2mn\|\mathcal{C}_k\|)$
UNIDDP	$O(k^2)$	$O(k^3m\|\mathcal{C}_k\|)$
NORMSDP	$O(k^2n)$	$O(k^3mn\|\mathcal{C}_k\|)$
NORMDDP	$O(k^3)$	$O(k^4m\|\mathcal{C}_k\|)$

6. EXPERIMENTS

In this section, we study the performance of the POPSM model and the PROBAPRI framework on both synthetic datasets and real datasets. For clarity in our result presentation, we denote the POPSM model by *U-POPSM* for the UniDist matrix and by *N-POPSM* for the NormDist matrix, and compare these two POPSM models with the OPSMRM model [Chui et al. 2008; Yip et al. 2013] and the OPSM model [Ben-Dor et al. 2002]. We also compare the POPSM mining methods with that of OPSMRM. By setting the number of replicates to be 1, we employ the OPSMRM mining method to mine OPSM patterns. We implement the more efficient HT-Bound as the pruning strategy for the OPSMRM mining method. All the algorithms are implemented in C++, and all the experiments are conducted on a Macbook Pro with 2.53GHZ CPU and 4G memory.

6.1. Data Preparation

Three real datasets are used in our experiments: a yeast galactose dataset [Ideker et al. 2001; Yeung et al. 2003] (or simply the yeast dataset), a rat circadian dataset [Nguyen et al. 2010] (or simply the rat dataset), and an RFID user trace dataset[1] (or simply the RFID dataset).

The yeast dataset contains 205 genes and 20 experimental conditions, and there are four replicates for each entry of the matrix. This set-valued matrix has been used for mining OPSMRM patterns. We generate a real-valued matrix by taking the average of the four replicates of an entry as the new single entry value, and the generated matrix is taken as input for mining OPSM patterns. We adopt two methods to convert the original set-valued matrix into a UniDist matrix. First, we set the range of every entry by taking the smallest and largest values of the four replicates as the lower and upper bounds, which we call the *MinMax* method. Second, we compute the mean μ and

[1]`http://lahar.cs.washington.edu/displayPage.php?path=./content/Download/RFIDData/rfidData.html`

the standard deviation σ based on the replicates of every entry and set the range to $[\mu - t * \sigma, \mu + t * \sigma]$ where t is set to 1, 2, or 3, and we call this the *Std* method. By using the *Std* method, we expect the range covers the distribution of true replicates and reduces the influence of possibly noisy ones. To generate the NormDist matrix, we compute the mean and standard deviation of an entry based on the replicates. During the cubic spline approximation, the approximation range is set to be $[\mu - t * \sigma, \mu + t * \sigma]$ and the range is divided into q pieces, where t is set to the values in $\{1, 2, 3\}$ and q is set to the values in $\{4, 6, 8, 10\}$.

The rat dataset contains 527 genes and 10 conditions, and it is extracted using two-sample t-test from a larger dataset, which is collected by examining the fluctuation of gene expressions in the rat livers within the 24 hour circadian cycle. It is also a set-valued matrix, and the number of replicates under a condition varies from 2 to 6. We adopt the same method used in the yeast dataset to convert the rat dataset into the UniDist matrix and the NormDist matrix.

The RFID dataset contains the traces of twelve voluntary users in a building with 195 RFID antennas installed. Only 44 antennas are active, that is, they detect the trace of at least one user for at least one time. The trace of a user consists of a series of (period, antenna) pairs which indicates the time period during which the user is continuously detected by the antenna. When a user is detected by the same antenna at different time periods, we split the raw trace into several new traces in which the user is detected by any antenna for at most one time. We then generate a UniDist matrix having 103 rows (i.e., user traces) and 44 columns (i.e., antennas). The set of timestamps within each time period is enumerated to generate the corresponding set-valued matrix, and the middle time points are computed to generate the real-valued matrix.

The synthetic datasets are generated based on the real yeast dataset. To generate an r-by-c matrix, we first randomly pick r rows and c columns from the original matrix and generate an r-by-c set-valued matrix. To simulate randomness, each entry in the matrix is shifted left or right with a small constant, or is kept unchanged with equal probability. Then, similar methods used for the real datasets are employed to convert the set-valued matrix into the UniDist or NormDist matrix.

6.2. Evaluation Measures

We now explain the methodology of evaluating the experimental results. There are three evaluation measures, namely the *biological significance* of the patterns mined from the biological datasets, the *trace accuracy score* of the patterns mined from the RFID dataset, and the *running time* for mining patterns in general.

6.2.1. Biological Significance. To compare the effectiveness of different models that promote the discovery of interesting patterns from gene expression data, we validate the *biological significance* of the POPSM, OPSMRM, and OPSM patterns mined from the two biological datasets. We adopt a measurement called p-value [Zhang et al. 2008; Fang et al. 2010; Fang et al. 2012], which is well accepted for measuring the association between mined patterns and the known gene functional categories. Essentially, a smaller p-value indicates a stronger association between a pattern and some gene category, and the pattern is thus regarded as more biologically significant.

6.2.2. Trace Accuracy. The patterns mined from the RFID dataset consists of a set of users sharing a common subroute passing a subset of antennas. To evaluate whether the subroute really matches the ground-truth traces of the users, we define a measure called the *trace accuracy score* and denote this score by $S(P, Q, \tau_Q)$ where $(P, Q : \tau_Q)$ is a POPSM, an OPSMRM or an OPSM pattern having τ_Q as the backbone order. The trace accuracy score of a pattern $(P, Q : \tau_Q)$ is computed as follows.

$$S(P, Q, \tau_Q) = \frac{1}{|P|} \sum_{g_i \in P} \frac{|LCS(\tau_Q, \mathcal{T}_{g_i})|}{|\tau_Q|},$$

where $\mathcal{T}_{g_i}$ is the ground-truth trace of user g_i, and $|LCS(\tau_Q, \mathcal{T}_{g_i})|$ is the length of the longest common subsequence between the detected subroute τ_Q and $\mathcal{T}_{g_i}$. The introduction of the two normalized factors $|P|$ and $|\tau_Q|$ in the score function makes the scores of patterns with different sizes comparable.

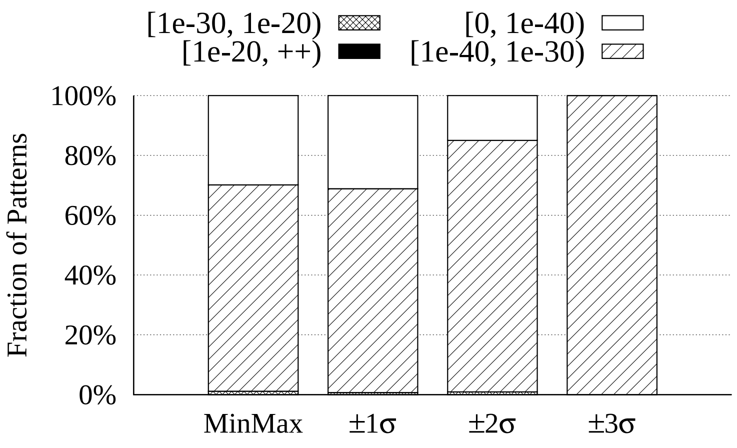
6.2.3. Efficiency. We study the efficiency of different methods by comparing their running times in two ways. First, we denote by *TR-Time* the total running time for generating the patterns. Second, we also study the average time for mining a single pattern, which is denoted by *SP-Time*, since the number of patterns mined by one method or under one setting is not the same as that mined by another method or under another setting in general.

6.3. Comparison of Data Conversion Methods

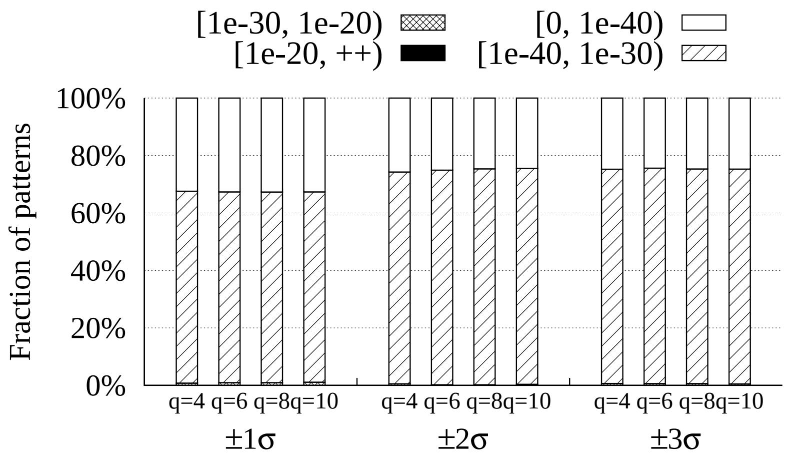
Given a set-valued matrix, we use the *MinMax* or *Std* methods described in Section 6.1 to convert it to a UniDist or a NormDist matrix. In the first set of experiments, we use the yeast dataset to study the impact of conversion methods on the quality of mined U-POPSM and N-POPSM patterns. The DDP technique is adopted for both the UNIAPRI and NORMAPRI methods.

When the *Std* method is taken for generating the UniDist matrix, the parameter t for the approximation range $[\mu - t * \sigma, \mu + t * \sigma]$ is additionally set to 1, 2, and 3. Including the UniDist matrix generated by *MinMax*, there are four UniDist matrices generated by different methods or under different settings. Based on each UniDist matrix, the UNIAPRI method is employed to mine U-POPSM patterns containing at least 60 rows and 5 columns, and having $\alpha = 0.3$. We set four p-value ranges which are $[1e-20, ++)$, $[1e-30, 1e-20)$, $[1e-40, 1e-30)$ and $[0, 1e-40)$, and call them *significance levels*. If the p-value of a pattern falls within any range, we say that the pattern reaches the corresponding significance level. Figure 8(a) shows the fractions of patterns that are mined from different UniDist matrices and reach different significance levels. The bar labeled "MinMax" in the figure corresponds to the statistics of the patterns mined from the UniDist matrix generated by the *MinMax* method, and the bar labeled "$\pm 1\sigma$" corresponds to the statistics of the patterns mined from the UniDist matrix generated by the *Std* method with $t = 1$. In all the series, the white color is used to show the fraction of the most significant patterns whose p-values fall within the highest significance level $[0, 1e-40)$.

Among the patterns mined under the settings "MinMax" and "$\pm 1\sigma$", about 30% of the patterns have the p-values smaller than $1e - 40$, which means that they reach the highest significance level $[0, 1e-40)$. In contrast, only 15% of the patterns mined under the setting "$\pm 2\sigma$" reach the same level, and no pattern mined under the setting "$\pm 3\sigma$" reaches this significance level. A possible explanation is that the approximation range $[\mu - t\sigma, \mu + t\sigma]$ with $t \geq 2$ is too large, which covers not only the true replicates but also the noisy ones. Under the uniform distribution model, both the true replicates and the noisy ones are associated with equally large probability densities, and thus the quality of mined patterns are affected due to the coverage of noisy replicates by a larger approximation range. Although the quality of the patterns mined under the setting "$\pm 1\sigma$" and that of the patterns mined under the setting "MinMax" are very similar, the fraction of patterns that are mined under the setting "$\pm 1\sigma$" and reach the highest significance level, i.e., 31.2%, is slightly larger than that of patterns that are mined under "MinMax" and reach the same level, i.e., 29.9%. Besides, slightly larger



(a) UniDist matrix



(b) NormDist matrix

Fig. 8. Comparison of matrix conversion methods

fraction of patterns mined under "MinMax" fall within the lower significance level $[1e-30, 1e-20)$. Thus, we generate the UniDist matrix by setting the approximation range to $[\mu-\sigma, \mu+\sigma]$ in the subsequent experiments.

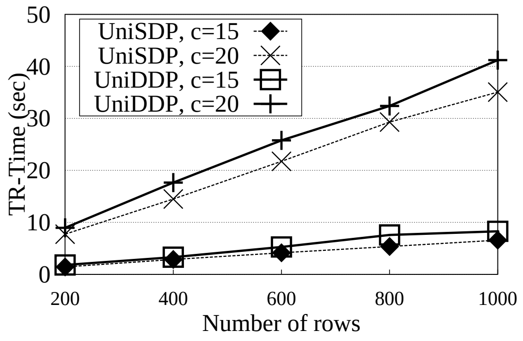
When the NORMAPRI method is adopted to mine N-POPSM patterns, we set the approximation range of the cubic spline to $[\mu - t*\sigma, \mu + t*\sigma]$ with t being equal to the values in $\{1, 2, 3\}$. In addition, the cubic spline technique interpolates a normal distribution with q polynomials, where q is set to the values in $\{4, 6, 8, 10\}$. For different combinations of t and q values, we adopt the NORMAPRI method to mine N-POPSM patterns that contain at least 60 rows and 5 columns and satisfy the support threshold $\alpha = 0.3$. The statistics on the quality of the mined patterns are shown in Figure 8(b), where the bar labeled "$\pm 1\sigma, q$=4" corresponds to the results with the approximation range $[\mu - \sigma, \mu + \sigma]$ and four intervals in the spline approximation. Again, comparing the three settings of the approximation range, a larger fraction of mined patterns reach the highest significance level when the approximation range is $[\mu - \sigma, \mu + \sigma]$. The reason is similar to that for the UniDist matrix: a larger approximation range is more likely to cover many noisy replicates, which accordingly influences the quality of the mined patterns. With the approximation range fixed, the setting of q does not influence the quality of the mined patterns. For example, when the approximation range is set to $[\mu - \sigma, \mu + \sigma]$, no matter what q value is, 32% of the patterns reach the highest significance level, and 1% of the patterns reach the significance level of $[1e - 30, 1e - 20)$. However, a larger q implies that a normal distribution is interpolated with more polynomials, which accordingly influences the efficiency of the NORMAPRI method. Therefore, we choose $[\mu - \sigma, \mu + \sigma]$ to be the approximation range for the NormDist matrix and always set q to 4 in the experiments discussed in subsequent sections.

Comparing these two types of the probabilistic matrices, the quality of the N-POPSM patterns mined from the NormDist matrix is less affected by the conversion methods and the settings. For example, when the approximation range is $[\mu - \sigma, \mu + \sigma]$, the percentage of the U-POPSM and N-POPSM patterns that reach the highest significance level are both around 30%. When the approximation range is $[\mu - 3\sigma, \mu + 3\sigma]$, there are still about 25% of the N-POPSM patterns reaching the highest significance level, while no U-POPSM pattern reaches the same level. The reason is that, although a larger approximation range may cover more noisy replicates in the NormDist matrix, these noisy values are usually farther away from the mean of the range, and so they are associated with very small probability densities. The influence of such noisy values to the computation of probabilistic support is then limited, and the quality of mined N-POPSM patterns can be well maintained. In the UniDist matrix, a larger approximation range also covers more noisy replicates, which however are associated with equally large probabilities as the true replicates. The inclusion of the noisy replicates affects the accuracy of the probabilistic supports of rows w.r.t. orders, and accordingly degrades the quality of mined U-POPSM patterns. Therefore, compared to the UniDist matrix, the NormDist matrix well smoothes out the influence of noisy replicates to the quality of mined patterns, and better models the noisy scientific data like the yeast dataset we are using. However, by carefully choosing the conversion method as well as the approximation range when generating the UniDist matrix, the quality of the U-POPSM patterns can still be maintained.

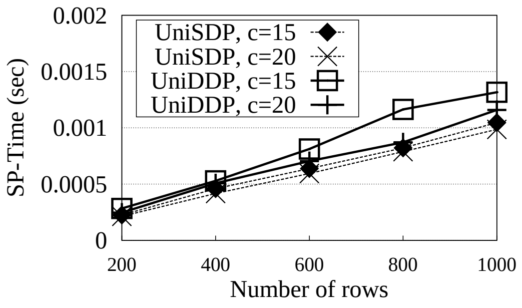
6.4. Scalability of PROBAPRI

In this set of experiments, we use the synthetic datasets to study the efficiency of the UNIAPRI and NORMAPRI methods w.r.t. the size of the input matrix. As both methods can adopt either the SDP or the DDP technique for candidate verification, we compare the TR-Time and SP-Time of each method using the two DP techniques. Following the convention introduced in Table III, we denote the UNIAPRI method with the static (or respectively the dynamic) DP technique by UNISDP (or respectively UNIDDP), and denote the NORMAPRI method with the static (or respectively the dynamic) DP technique by NORMSDP (or respectively NORMDDP).

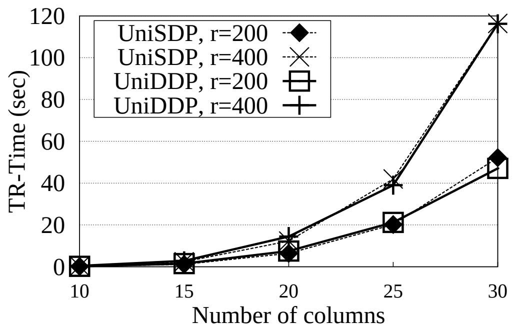
First, we set the number of columns c to 15 and 20, vary the number of rows with the values in $\{200, 400, 600, 800, 1000\}$, and adopt the *Std* method described in Section 6.1 to generate r-by-c UniDist matrices. The approximation range is set to $[\mu - \sigma, \mu + \sigma]$.



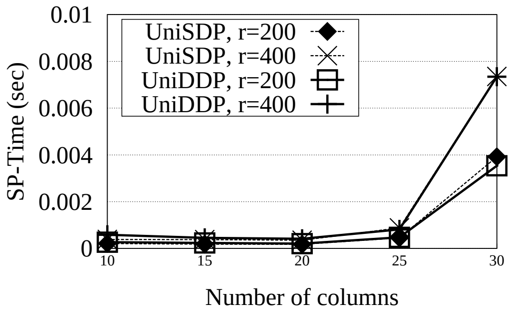
(a) TR-Time w.r.t. r



(b) SP-Time w.r.t. r



(c) TR-Time w.r.t. c

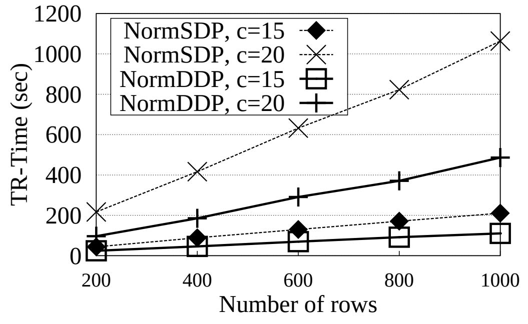


(d) SP-Time w.r.t. c

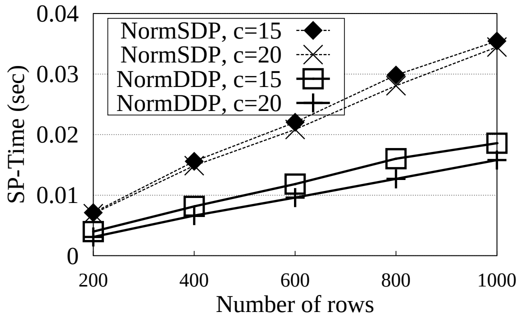
Fig. 9. Running time w.r.t. the size of a UniDist matrix

Then, the UNIAPRI method is adopted to mine U-POPSM patterns. The size threshold r_{min} is set to $10\% \times r$, and c_{min} is varied from 4 to 8 as the number of columns increases from 10 to 30 in order to control the number of patterns. The support threshold α is fixed to 0.3. Figures 9(a) and 9(b) show the TR-Time and SP-Time of UNISDP and UNIDDP w.r.t. the number of rows in the input UniDist matrix. Then, we set the number of rows r to 200 and 400, and generate the UniDist matrices by varying the number of columns with the values in $\{10, 15, 20, 25, 30\}$. Figures 9(c) and 9(d) show the TR-Time and SP-Time of UNISDP and UNIDDP w.r.t. the number of columns in the input matrix.

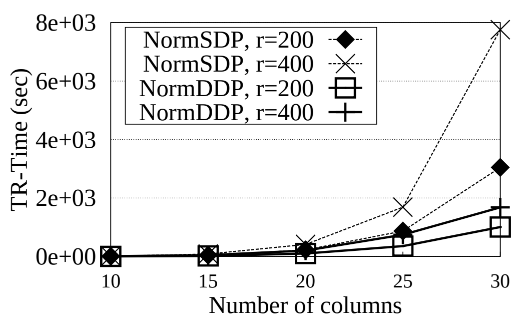
From Figure 9(a), we find that both the UNISDP and the UNIDDP methods scale well w.r.t. the number of rows in the UniDist matrix. Specifically, when the number of columns c is fixed to 15 and the number of rows increases by 5 times, from 200 to 1000, the TR-time increases linearly, from 1.4 to 6.6 seconds for UNISDP and from 1.8 to 8.3 seconds for UNIDDP. The same trend is observed again in Figure 9(b), where the SP-Time of the two methods is compared. Notably, UNISDP and UNIDDP are very efficient in mining U-POPSM patterns, and both take less than 0.0015 seconds for mining a pattern from a UniDist matrix having 1000 rows and 20 columns. In comparison, these two methods are more affected by the number of columns in the



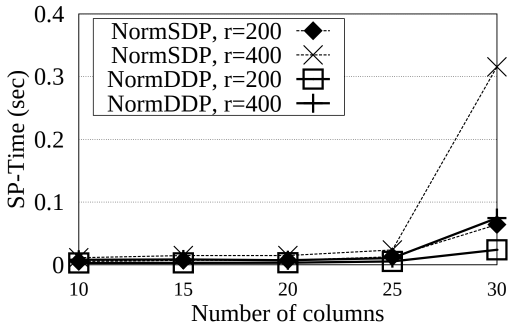
(a) TR-Time w.r.t. r



(b) SP-Time w.r.t. r



(c) TR-Time w.r.t. c



(d) SP-Time w.r.t. c

Fig. 10. Running time w.r.t. the size of NormDist matrix

input matrix, as shown in Figures 9(c) and 9(d). The reason is that, an increase of the number of columns results in an increase of the number of candidate orders during the candidate generation step and, thus, more time is needed to generate and verify the candidate orders.

Comparing UNISDP and UNIDDP, we find that UNISDP is slightly more efficient than UNIDDP. The advantage of UNISDP is more apparent when the number of columns in the matrix is smaller. The reason is that, the DP process needed for candidate verification is already very efficient for the UniDist matrix. Although the DDP technique adopted by UNIDDP is designed to reduce the size of the DP table and lower the time cost incurred by the DP process, dynamically updating the DP table takes extra time. Therefore, not much time can be saved eventually. However, when the number of columns in the input matrix is large enough, the DDP technique is beneficial. For example, when r equals 200 and c equals 30, the TR-Time and SP-Time of UNISDP are respectively 52.17 and 0.0039 seconds, while those of UNIDDP are respectively 47.06 and 0.0035 seconds. Since the performance of these two versions are comparable and both are efficient, we choose the DDP version of UNIAPRI in the subsequent experiments.

To study the scalability of the NORMAPRI method, we adopt the *Std* method to generate the NormDist matrix with settings $t = 1$ and $q = 4$. The number of rows r and the number of columns c are varied in the same way as they are for generating the UniDist matrix. When the N-POPSM patterns are mined, the size thresholds r_{min} is set to $10\% \times r$, c_{min} is varied from 4 to 8, and the support threshold α is fixed to 0.3. Figure 10 shows the TR-Time and SP-Time of NORMSDP and NORMDDP w.r.t. the size of the input NormDist matrix.

Similarly to the two versions of UNIAPRI, both NORMSDP and NORMDDP scale well as the number of rows in the input matrix increases, and they are more affected by the number of columns. Referring to Figures 10(a) and 10(b), we can see that, when the number of rows increases from 200 to 1000, the TR-Time and SP-Time of NORMSDP and NORMDDP all linearly increase by about 5 times. However, Figure 10(d) shows that, as the number of columns increases from 10 to 30, the SP-Time of NORMSDP increases by about 30 times while the SP-Time of NORMDDP increases by about 10 times. As the DP process in NORMAPRI is time-consuming, the adoption of the DDP technique greatly improves the efficiency of mining N-POPSM patterns, and the advantage becomes more apparent when r or c gets larger. When c is 30, the TR-Time of NORMDDP is 33% of the TR-Time of NORMSDP with $r = 20$ and is only 22% with $r = 40$. The saving of the SP-Time is also up to 70%.

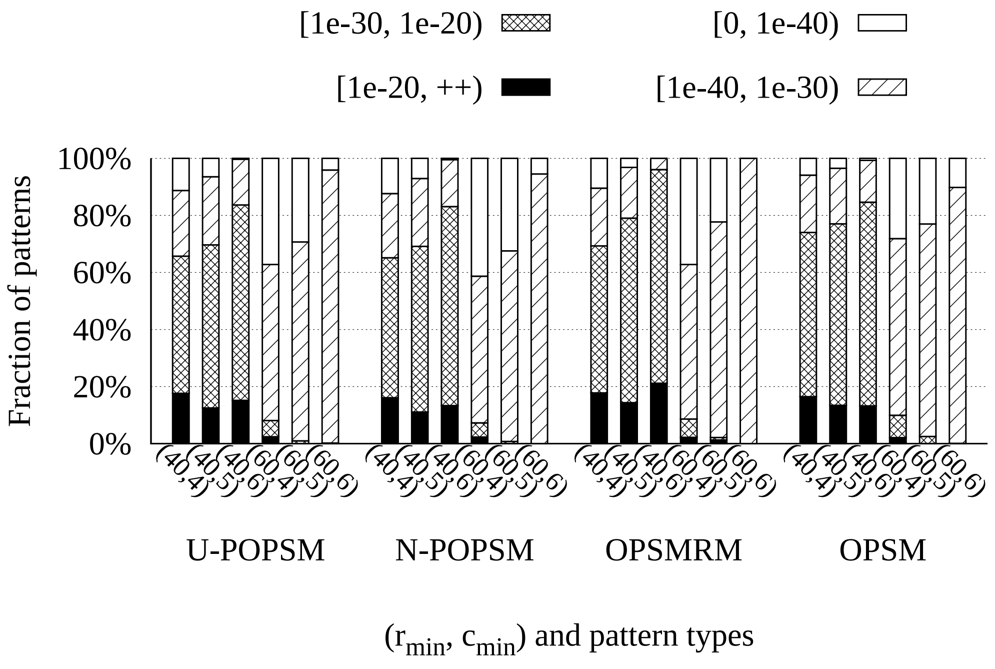
Note that, no matter which DP technique is taken in the UNIAPRI method, the number of patterns mined from the same UniDist matrix is the same. However, due to the computation error of cubic polynomial integral, the number of patterns mined by NORMAPRI using different DP techniques is slightly different. The difference is less than 0.3%, that is, the number of patterns differs by 3 for every 1000 patterns. Since the difference is not significant, we adopt the more efficient NORMDDP in NORMAPRI in the subsequent experiments.

6.5. Experiments on the Yeast Dataset

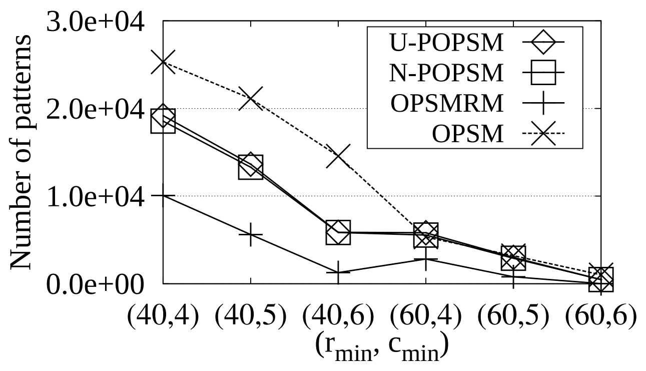
In this set of experiments, we use the yeast dataset to study the effectiveness of the POPSM model for discovering significant patterns. We study the settings of the parameters, such as the size thresholds r_{min} and c_{min}, and the support threshold α. The biological significance of the U-POPSM and N-POPSM patterns is compared with that of the OPSMRM and OPSM patterns. The efficiency of the corresponding mining methods is also studied.

6.5.1. Influence of the size thresholds. For real datasets, it is difficult to determine proper values for the two size thresholds r_{min} and c_{min} such that the patterns satisfying the thresholds are all guaranteed to have good quality. Ben-Dor et al. brought forward a heuristic solution to this problem [Ben-Dor et al. 2002]. Since patterns with larger size are statistically less likely to exist by random in a data matrix, these statistically significant patterns are more probable to reveal important knowledge hidden in the matrix, and thus should also be biologically significant. Therefore, in order to guarantee the quality of mined patterns, we prefer patterns with larger size. Suppose there is a random real-valued matrix having 205 rows and 20 columns, which is the same size as the yeast dataset. Since the number of possible orders of 3 columns is $3! = 6$, we should always be able to find an OPSM pattern with 3 columns and $\frac{205}{6} \approx 34$ rows, and so this pattern is of low statistical significance. To guarantee the quality of the patterns mined from the yeast dataset, we set the size thresholds to be larger than 34 rows and 3 columns.

We study the influence of the size thresholds to the quality of the patterns by fixing α to 0.3, and varying r_{min} with the values in $\{40, 50, 60, 70\}$ and varying c_{min} with the values in $\{4, 5, 6\}$. Due to space limit, we only present the results with r_{min} equal to



(a) Fraction of patterns w.r.t. the size thresholds



(b) Number of patterns w.r.t. the size thresholds

Fig. 11. Influence of the size thresholds c_{min} and r_{min}

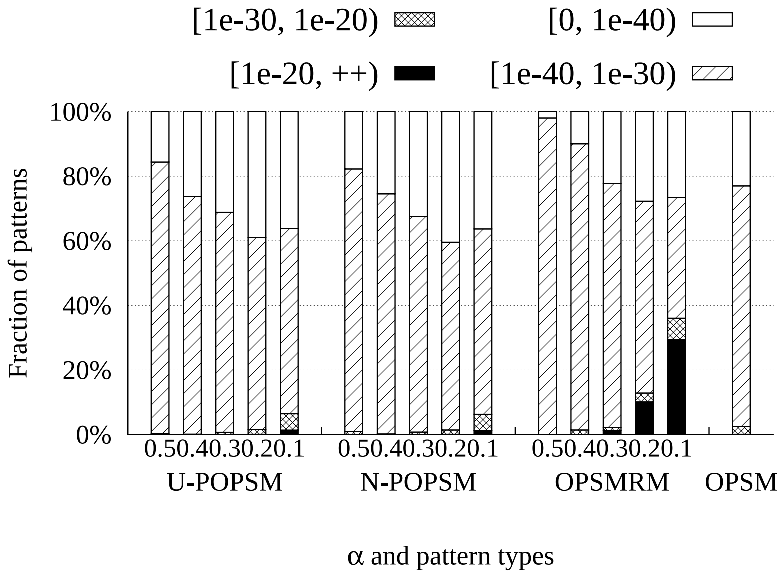
40 and 60. Similarly to the first set of experiments, we also set four significance levels. Figure 11(a) shows the fraction of patterns that respectively reach each significance level. The first bar "(40, 4), U-POPSM" corresponds to the distribution of the U-POPSM patterns mined by UNIAPRI with r_{min} and c_{min} respectively set to 40 and 4. We can

see that, when r_{min} is smaller than or equal to 50, for all the four pattern types, there are around 40–90% of the mined patterns falling within the two lowest significance levels. The quality of the mined patterns however improves substantially when r_{min} is larger than or equal to 60. Especially when c_{min} is larger than or equal to 5, less than 1% of the U-POPSM and N-POPSM paterns, about 2% of the OPSMRM patterns, and 2.5% of the OPSM patterns fall within the two lowest significance levels. On the other hand, the number of mined patterns decreases a lot when r_{min} and c_{min} get larger, as shown in Figure 11(b). When r_{min} equals 60 and c_{min} equals 6, around 470 U-POPSM and N-POPSM patterns have been mined, but only 11 OPSMRM patterns have been mined. To achieve a good balance between pattern quality and pattern number, we therefore respectively set r_{min} and c_{min} to 60 and 5 in the experiments.

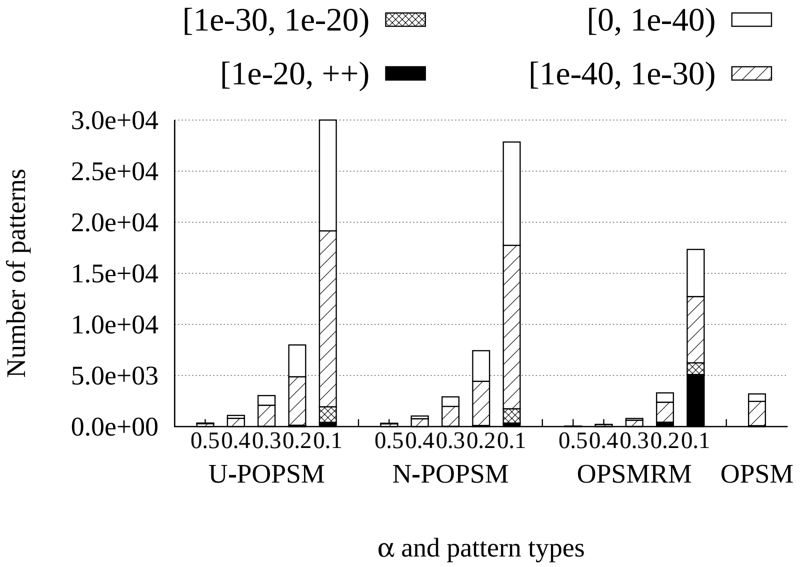
6.5.2. Influence of the support threshold. As a larger probabilistic support (or fractional support used by the OPSMRM model) value intuitively implies a better or more persuasive support of a row w.r.t. an order. When mining the POPSM or OPSMRM patterns, a support threshold α is needed to exclude those rows whose support to a certain order is too small to be meaningful. Similarly to the setting of the size thresholds, the setting of α should ideally be determined by the noise level of the datasets. However, such prior knowledge about the real datasets is usually not available. Therefore, we empirically study the influence of the support threshold to the quality of mined patterns.

We fix the size thresholds r_{min} and c_{min} to 60 and 5, and vary α from 0.1 to 0.5 for UNIAPRI, NORMAPRI, and the OPSMRM mining method. The OPSMRM mining method is also employed for mining the OPSM patterns with $\alpha = 1$. The statistics on the biological significance of the patterns are shown in Figures 12(a) and 12(b). Figure 12(a) shows the fractions of patterns that reach the four significance level. For all U-POPSM, N-POPSM, and OPSMRM patterns, the fraction of the most significant patterns generally increases as α decreases, and the best result for each model is obtained when α equals 0.2. With the same α value, the fractions of the most significant U-POPSM and N-POPSM patterns are apparently larger than that of the most significant OPSMRM patterns, and the fraction of such N-POPSM patterns is slightly larger than that of the U-POPSM patterns. On the other hand, less than 0.06% of U-POPSM and N-POPSM patterns fall within the lowest significance level, while about 10% of OPSMRM patterns fall within this level. If we consider the absolute number of patterns as shown in Figure 12(b), the advantage of the U-POPSM and N-POPSM models becomes very clearly. When α is 0.2, 3112 U-POPSM patterns and 3004 N-POPSM patterns reach the highest significance level, while only 915 OPSMRM patterns and 735 OPSM patterns reach the same level.

We observe that PROBAPRI mines a large fraction of the most significant POPSM patterns but still keeps the fraction of the least significant ones small. This fact again confirms the superiority of our probabilistic matrix representation compared to the set-valued matrix representation. In the set-valued matrix, a small setting of α like 0.2 promotes the discovery of the most significant patterns. However, the noisy replicates, which have an equally large probability as the true replicates, may result in that uncorrelated rows also satisfy the small threshold and thus are included in many patterns. The quality of these patterns are degraded, and the fraction of the least significant patterns accordingly becomes larger. In contrast, when we convert the yeast dataset to a NormDist matrix, noisy replicates that are usually far away from the mean of the approximation range are associated with small probability densities, and thus their effect to the computation of the probabilistic support is limited. As the probabilistic support based on the NormDist matrix more accurately reflects the supports of rows w.r.t. orders, the quality of mined N-POPSM patterns are well maintained. By



(a) Fraction of patterns at four significance levels

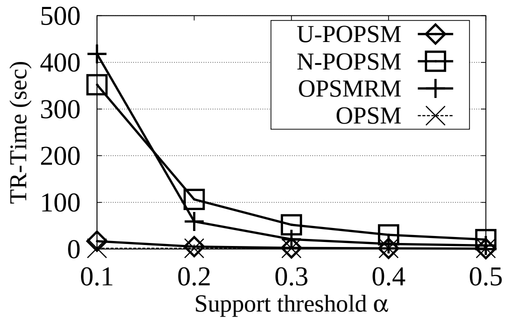


(b) Number of patterns at four significance levels

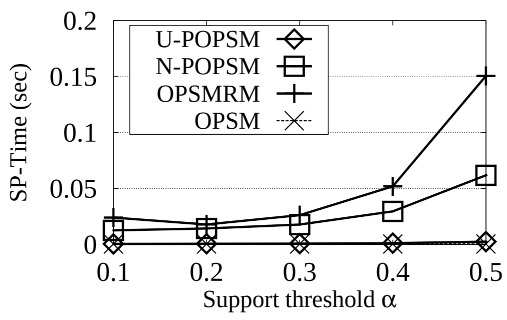
Fig. 12. Distribution of the p-values of the patterns mined from the yeast data

carefully choosing the conversion method as well as the approximation range of the UniDist matrix, the influence of noisy replicates can be eliminated and the quality of mined U-POPSM pattern are still well controlled.

Let us illustrate the biological significance of mined POPSM patterns with a specific case. A U-POPSM pattern is mined when α is 0.2, which contains 82 genes and gets



(a) TR-Time w.r.t. α



(b) SP-Time w.r.t. α

Fig. 13. Running time w.r.t. the support threshold

the smallest p-value $3.99e-61$. All of the 82 genes in the pattern are annotated with the same functional *Gene Ontology*[2] (GO) term, and this pattern exactly covers all the genes that are annotated with this functional GO term in the input data.

Next, we compare the efficiency of different mining methods. Figures 13(a) and 13(b) respectively show the TR-Time and the SP-Time for mining the four types of patterns. Referring to Figure 13(a), we can see that the TR-Time for mining U-POPSM, N-POPSM, and OPSMRM patterns all decreases as α increases, since fewer patterns are mined by each method. Because the number of patterns mined by each method is different, we compare the efficiency in terms of the SP-Time, as shown in Figure 13(b), in order to gain better insight of scalability. The SP-Time of U-POPSM is much shorter than that of OPSMRM for all α values. The SP-Time of N-POPSM is comparable to that of OPSMRM when α is small, and is better controlled when α increases from 0.1 to 0.5. This set of results again demonstrates the superiority of the DDP technique for improving the efficiency of the NORMAPRI method. Besides, the combination of the *CandTree* and the DP process makes the candidate verification process more efficient. Thus, the SP-Time of POPSM patterns can be better controlled.

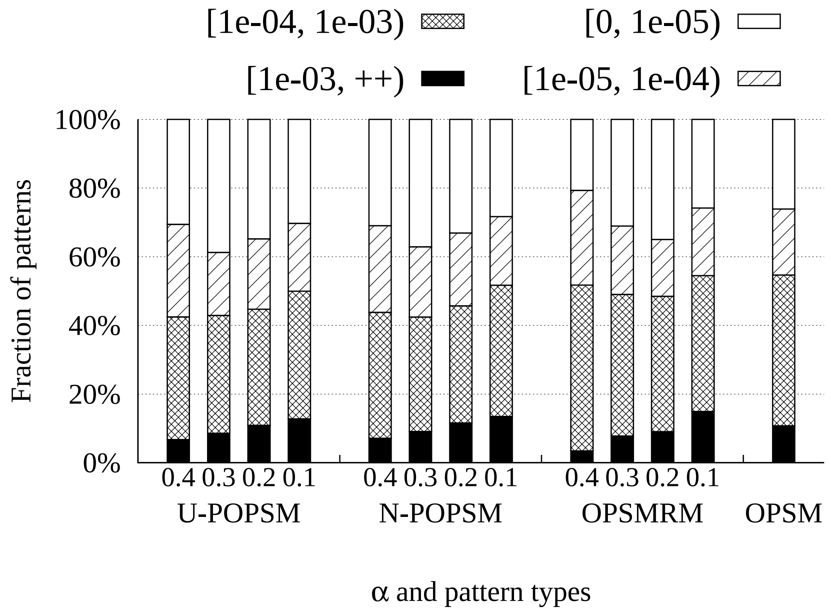
From this set of experiments, we can see that, based on the converted UniDist and NormDist matrices, the POPSM model better promotes the discovery of more significant patterns compared with the OPSMRM and OPSM models, in the sense that a larger number and fraction of the most significant patterns are discovered. Comparing the U-POPSM model and the N-POPSM model, the fraction of N-POPSM patterns that reach the highest significance level is slightly larger than that of U-POPSM reaching the same level, although the most significant patterns of both types are similar in number.

6.6. Experiments on the Rat Dataset

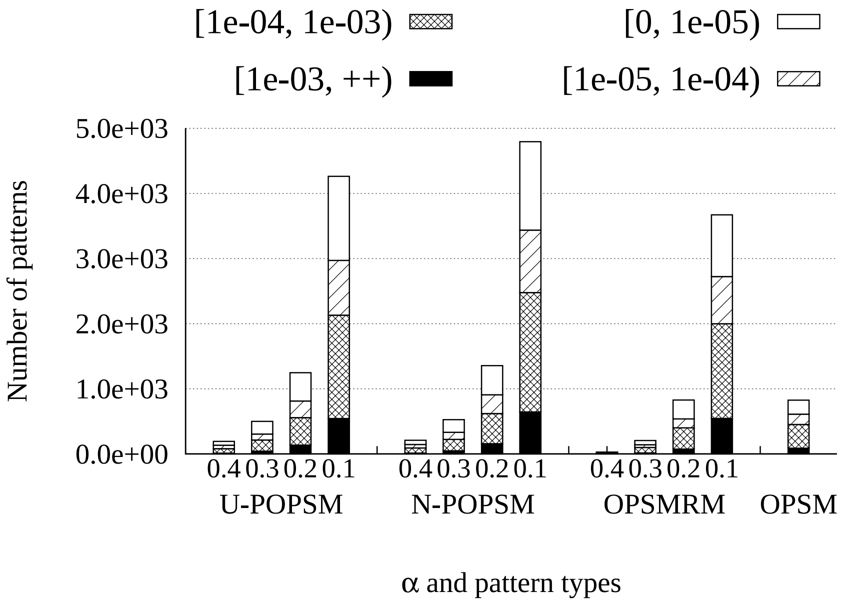
We use another real biological dataset, i.e., the rat dataset, to further study the effectiveness of the U-POPSM and N-POPSM models in identifying significant patterns.

We omit the study of the influence of the size thresholds for this dataset and simply fix r_{min} and c_{min} to 50 and 4. The support threshold α is varied from 0.1 to 0.4. We set four significance levels $[1e-3,++)$, $[1e-4,1e-3)$, $[1e-5,1e-4)$, and $[0,1e-5)$. Figures 14(a) and 14(b) show, respectively, the fraction and number of different types

[2]http://www.geneontology.org/



(a) Fraction of patterns at four significance levels



(b) Number of patterns at four significance levels

Fig. 14. Distribution of the p-values of the patterns mined from the rat data

of patterns that reach each significance level. When the U-POPSM model or the N-POPSM model is adopted, the fraction of the most significant patterns reaches the maximum value when α equals 0.3, and when the OPSMRM model is adopted, the

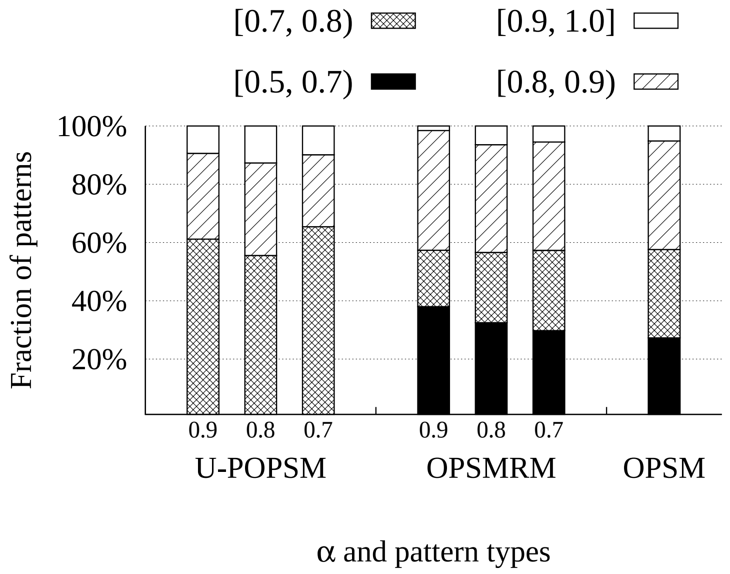


Fig. 15. Distribution of the accuracy score of the patterns mined from the RFID dataset

fraction of the most significant patterns reaches the maximum value when α equals 0.2. Under the best setting for each model, larger fractions of U-POPSM and N-POPSM patterns reach the highest significance level compared to the OPSMRM and OPSM patterns. In addition, the fraction of the U-POPSM patterns that fall within the lowest significance level is the smallest among the four models. If the number of patterns that reach the highest significance level is compared, both the U-POPSM and N-POPSM models outperform the OPSMRM model for all α values as well.

6.7. Experiments on the RFID Dataset

In the last set of experiments, we use the RFID dataset to evaluate the capability of the POPSM model in revealing the common subroutes of the users. Since the RFID dataset is given as a UniDist matrix, we only study the effectiveness of the U-POPSM model. We convert the UniDist matrix to a set-valued matrix by enumerating the timestamps within each range. We also convert the UniDist matrix to a real-valued matrix by taking the middle time point of each range. We set r_{min} and c_{min} to 12 and 9, and mine U-POPSM, OPSMRM and OPSM patterns. Figure 15 shows the distribution of the accuracy scores of the patterns. For a particular pattern, the larger its score is, the more accurate it is at capturing the common subroute of the set of involved users. When mining U-POPSM and OPSMRM patterns, we vary α from 0.7 to 0.9. In terms of the fraction of patterns that fall into the highest score range $[0.9, 1.0]$, both methods achieve the best result when α equals 0.8. The reason may be that, a large setting of α, say 0.8, would be too strict to tolerate intrinsic noise in RFID data, while setting α with a smaller value would obscure the true traces of the users. From Figure 15, we can see that, compared to the OPSMRM and OPSM patterns, a larger fraction of U-POPSM patterns fall into the highest score range. Furthermore, the accuracy scores of all the U-POPSM patterns are above 0.7, while more than 25% of the OPSMRM and OPSM patterns have the accuracy scores below 0.7.

7. CONCLUSIONS

In this paper, we study the problem of mining probabilistic OPSM patterns from uncertain data matrices having continuous distributions. We consider two representative distributions, the uniform distribution and the normal distribution, and accordingly formulate the uncertain data as the UniDist matrix and the NormDist matrix. Compared to a set-valued matrix representation, our UniDist and NormDist matrix representations better smooth out the influence of noise in real datasets. Based on the UniDist and NormDist matrices, we propose a novel probabilistic OPSM (POPSM) model. We define a new probabilistic support measure that evaluates the extent to which a row supports an order. The POPSM model uses a backbone order to capture a consensus trend, and all the rows in a POPSM pattern are required to support the backbone order of the pattern with sufficiently large probabilistic support.

In order to mine POPSM patterns effectively and efficiently, we develop two new POPSM mining methods: UNIAPRI for the UniDist matrix and NORMAPRI for the NormDist matrix. In both UNIAPRI and NORMAPRI, a prefix-tree structure called *CandTree* is designed to organize the intermediate patterns in a compact way. Two novel dynamic programming techniques, the static DP and the dynamic DP, are also exploited in our methods, and the DP computation process is interwoven with the traversal of *CandTree* for efficient pattern verification.

We evaluate our POPSM mining approach on two real biological datasets, one real RFID dataset, and several synthetic datasets. Our experiments on the biological datasets show that, comparing with the counterpart OPSMRM model, the POPSM model better captures the characteristics of the expression levels of biologically correlated genes, and greatly promotes the discovery of patterns with high biological significance. Through the experiments on the RFID data, the POPSM model is shown to be capable of accuratly discovering users' common subroutes from noisy trace data. The experiments on the synthetic datasets demonstrate the efficiency of the UNIAPRI method. In particular, the dynamic DP technique greatly improves the efficiency of the NORMAPRI method.

Finally, it is worth noting that, while we study in depth two representative probabilistic distributions of uncertain data in this work, the spline technique adopted by NORMAPRI is flexible. By using appropriate spline interpolation, NORMAPRI can be adapted to dealing with other probabilistic matrices that have more general continuous distributions with high accuracy.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their insightful comments.

REFERENCES

AGGARWAL, C. C., LI, Y., WANG, J., AND WANG, J. 2009. Frequent pattern mining with uncertain data. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '09. ACM, New York, NY, USA, 29–37.

AGGARWAL, C. C., WOLF, J. L., YU, P. S., PROCOPIUC, C., AND PARK, J. S. 1999. Fast algorithms for projected clustering. In *Proceedings of the ACM SIGMOD international conference on Management of data*. SIGMOD '99. ACM, New York, NY, USA, 61–72.

AGRAWAL, R., GEHRKE, J., GUNOPULOS, D., AND RAGHAVAN, P. 1998. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the ACM SIGMOD international conference on Management of data*. SIGMOD '98. ACM, New York, NY, USA, 94–105.

AGRAWAL, R. AND SRIKANT, R. 1995. Mining sequential patterns. In *Proceedings of the IEEE 11th International Conference on Data Engineering*. ICDE '95. IEEE Computer Society, Washington, DC, USA, 3–14.

AHSANULLAH, M., NEVZOROV, V., AND SHAKIL, M. 2013. *An Introduction to Order Statistics*. Atlantis Studies in Probability and Statistics Series, vol. 3. Atlantis Press.

BANERJEE, A., DHILLON, I., GHOSH, J., MERUGU, S., AND MODHA, D. S. 2004. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. In *Proceedings of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '04. ACM, New York, NY, USA, 509–514.

BEN-DOR, A., CHOR, B., KARP, R., AND YAKHINI, Z. 2002. Discovering local structure in gene expression data: the order-preserving submatrix problem. In *Proceedings of the sixth Annual International Conference on Research in Computational Molecular Biology*. 49–57.

BUSYGIN, S., JACOBSEN, G., KRAMER, E., AND AG, C. 2002. Double conjugated clustering applied to leukemia microarray data. In *the Second SIAM ICDM Workshop on clustering high dimensional data*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

CHENG, Y. AND CHURCH, G. M. 2000. Biclustering of expression data. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*. 93–103.

CHEUNG, L., YIP, K. Y., CHEUNG, D. W., KAO, B., AND NG, M. K. 2007. On mining micro-array data by order-preserving submatrix. *International Journal of Bioinformatics Research and Applications 3,* 1, 42–64.

CHIA, B. K. H. AND KARUTURI, R. K. M. 2010. Differential co-expression framework to quantify goodness of biclusters and compare biclustering algorithms. *5,* 23.

CHO, H., DHILLON, I. S., GUAN, Y., AND SRA, S. 2004. Minimum sum-squared residue co-clustering of gene expression data. In *Proceedings of the SIAM international conference on Data Mining*. SDM '04. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 114–125.

CHUI, C. K., KAO, B., YIP, K. Y., AND LEE, S. D. 2008. Mining order-preserving submatrices from data with repeated measurements. In *Proceedings of the Eighth IEEE International Conference on Data Mining*. ICDM '08. IEEE Computer Society, Washington, DC, USA, 133–142.

DARURU, S., MARÍN, N., WALKER, M., AND GHOSH, J. 2009. Pervasive parallelism in data mining: Dataflow solution to co-clustering large and sparse netflix data. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '09. ACM, New York, NY, USA, 1115–1123.

DHILLON, I. S., MALLELA, S., AND MODHA, D. S. 2003. Information-theoretic co-clustering. In *Proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '03. ACM, New York, NY, USA, 89–98.

DING, C., LI, T., PENG, W., AND PARK, H. 2006. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '06. ACM, New York, NY, USA, 126–135.

FANG, Q., NG, W., AND FENG, J. 2010. Discovering significant relaxed order-preserving submatrices. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '10. ACM, New York, NY, USA, 433–442.

FANG, Q., NG, W., FENG, J., AND LI, Y. 2012. Mining bucket order-preserving submatrices in gene expression data. *IEEE Transactions on Knowledge and Data Engineering 24,* 12, 2218–2231.

GAO, B. J., GRIFFITH, O. L., ESTER, M., AND JONES, S. J. M. 2006. Discovering significant opsm subspace clusters in massive gene expression data. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '06. ACM, New York, NY, USA, 922–928.

GAO, B. J., GRIFFITH, O. L., ESTER, M., XIONG, H., ZHAO, Q., AND JONES, S. J. M. 2012. On the deep order-preserving submatrix problem: A best effort approach. *IEEE Transactions on Knowledge and Data Engineering 24,* 2, 309–325.

GETZ, G., LEVINE, E., AND DOMANY, E. 2000. Coupled two-way clustering analysis of gene microarray data. *Proceedings of the National Academy of Sciences, USA 97,* 22, 12079–12084.

GU, T., WANG, L., WU, Z., TAO, X., AND LU, J. 2011. A pattern mining approach to sensor-based human activity recognition. *IEEE Transactions on Knowledge and Data Engineering 23,* 9, 1359–1372.

GÜNNEMANN, S., FÄRBER, I., VIROCHSIRI, K., AND SEIDL, T. 2012. Subspace correlation clustering: finding locally correlated dimensions in subspace projections of the data. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '12. ACM, New York, NY, USA, 352–360.

GUPTA, N. AND AGGARWAL, S. 2010. MIB: Using mutual information for biclustering gene expression data. *Pattern Recognition 43,* 8, 2692–2697.

GUPTA, R., RAO, N., AND KUMAR, V. 2010. Discovery of error-tolerant biclusters from noisy gene expression data. In *Proceedings of the Ninth International Workshop on Data Mining in Bioinformatics*. BIOKDD '10. ACM, New York, NY, USA.

HARTIGAN, J. A. 1972. Direct clustering of a data matrix. *Journal of the American Statistical Association 67,* 337.

HEATH, M. T. 2002. *Scientific computing: an introductory survey*. McGraw-Hill Higher Education.

HUGHES, T. R., MARTON, M. J., JONES, A. R., AND ET AL. 2000. Functional discovery via a compendium of expression profiles. *Cell 102*, 109–126.

HUMRICH, J., GARTNER, T., AND GARRIGA, G. C. 2011. A fixed parameter tractable integer program for finding the maximum order preserving submatrix. In *Proceedings of the 11th IEEE International Conference on Data Mining*. ICDM '11. IEEE Computer Society, Washington, DC, USA, 1098–1103.

IDEKER, T., THORSSON, V., RANISH, J. A., AND ET AL. 2001. Integrated genomic and proteomic analyses of a systematically perturbed metabolic network. *Science 292,* 5518, 929–934.

IHMELS, J., FRIEDLANDER, G., BERGMANN, S., SARIG, O., ZIV, Y., AND BARKAI, N. 2002. Revealing modular organization in the yeast transcriptional network. *Nature Genetics 31,* 4, 370–377.

JI, S., ZHANG, W., AND LIU, J. 2012. A sparsity-inducing formulation for evolutionary co-clustering. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '12. ACM, New York, NY, USA, 334–342.

KAILING, K., KRIEGEL, H.-P., AND KRÖGER, P. 2004. Density-connected subspace clustering for high-dimensional data. In *Proceedings of the fourth SIAM international conference on Data Mining*. SDM '04. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 246–257.

KRIEGEL, H.-P., KRÖGER, P., AND ZIMEK, A. 2009. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data 3,* 1, 1–58.

KUM, H.-C., PEI, J., WANG, W., AND DUNCAN, D. 2003. ApproxMAP: Approximate mining of consensus sequential patterns. In *Proceedings of the third SIAM international conference on Data Mining*. SDM '02. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 311–315.

LEE, M.-L. T., KUO, F. C., WHITMOREI, G. A., AND SKLAR, J. 2000. Importance of replication in microarray gene expression studies: Statistical methods and evidence from repetitive cdna hybridizations. *PNAS 97,* 18, 9834–9839.

LI, G., MA, Q., TANG, H., PATERSON, A., AND XU, Y. 2009. QUBIC: a qualitative biclustering algorithm for analyses of gene expression data. *Nucleic Acids Research 37,* 15, e101.

LI, J. AND DESHPANDE, A. 2010. Ranking continuous probabilistic datasets. *The Proceedings of the VLDB Endowment 3,* 1, 638–649.

LIU, J. AND WANG, W. 2003. OP-Cluster: Clustering by tendency in high dimensional space. In *Proceedings of the 3rd IEEE International Conference on Data Mining*. ICDM '03. IEEE Computer Society, Washington, DC, USA, 187–194.

LONG, B., ZHANG, Z., AND YU, P. S. 2005. Co-clustering by block value decomposition. In *Proceedings of the 11th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '05. ACM, New York, NY, USA, 635–640.

MADEIRA, S., TEIXEIRA, M., SA-CORREIA, I., AND OLIVEIRA, A. 2010. Identification of regulatory modules in time series gene expression data using a linear time biclustering algorithm. *IEEE/ACM Transactions on Computational Biology and Bioinformatics 7,* 1, 153–165.

MADEIRA, S. C. AND OLIVEIRA, A. L. 2004. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics 1,* 1, 24–45.

MOISE, G. AND SANDER, J. 2008. Finding non-redundant, statistically significant regions in high dimensional data: a novel approach to projected and subspace clustering. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '08. ACM, New York, NY, USA, 533–541.

MURALI, T. M. AND KASIF, S. 2003. Extracting conserved gene expression motifs from gene expression data. In *Pacific Symposium on Biocomputing*. 77–88.

MUZAMMAL, M. AND RAMAN, R. 2011. Mining sequential patterns from probabilistic databases. In *Proceedings of the 15th Pacific-Asia conference on Advances in knowledge discovery and data mining - Volume Part II*. PAKDD '11. Springer-Verlag, Berlin, Heidelberg, 210–221.

NGUYEN, T. T., ALMON, R. R., DUBOIS, D. C., JUSKO, W. J., AND ANDROULAKIS, I. P. 2010. Importance of replication in analyzing time-series gene expression data: Corticosteroid dynamics and circadian patterns in rat liver. *BMC Bioinformatics 11,* 279.

PAN, F., ZHANG, X., AND WANG, W. 2008. CRD: Fast co-clustering on large datasets utilizing sampling-based matrix decomposition. In *Proceedings of the ACM SIGMOD international conference on Management of data*. SIGMOD '08. ACM, New York, NY, USA, 173–184.

PANDEY, G., ATLURI, G., STEINBACH, M., MYERS, C. L., AND KUMAR, V. 2009. An association analysis approach to biclustering. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '09. ACM, New York, NY, USA, 677–686.

PARSONS, L., HAQUE, E., AND LIU, H. 2004. Subspace clustering for high dimensional data: a review. *SIGKDD Explore Newsletter 6*, 90–105.

PEI, J., HAN, J., MORTAZAVI-ASL, B., PINTO, H., AND ET AL. 2001. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of the IEEE 17th International Conference on Data Engineering*. ICDE '01. IEEE Computer Society, Washington, DC, USA, 215–224.

PEI, J., HAN, J., MORTAZAVI-ASL, B., WANG, J., AND ET AL. 2004. Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Transactions on Knowledge and Data Engineering 16*, 1424–1440.

PONTES, B., DIVINA, F., GIRÁLDEZ, R., AND ET AL. 2010. Improved biclustering on expression data through overlapping control. *International Journal of Intelligent Computing and Cybernetics 3*, 293–309.

PRELIĆ, A., BLEULER, S., ZIMMERMANN, P., WILLE, A., AND ET AL. 2006. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics 22,* 9, 1122–1129.

RASHIDI, P., COOK, D. J., HOLDER, L. B., AND SCHMITTER-EDGECOMBE, M. 2011. Discovering activities to recognize and track in a smart environment. *IEEE Transactions on Knowledge and Data Engineering 23,* 4, 527–539.

RÉ, C., LETCHNER, J., BALAZINKSA, M., AND SUCIU, D. 2008. Event queries on correlated probabilistic streams. In *Proceedings of the ACM SIGMOD international conference on Management of data*. SIGMOD '08. ACM, New York, NY, USA, 715–728.

SEIDEL, C. 2008. *Introduction to DNA Microarrays*. Wiley-VCH Verlag GmbH & Co. KGaA, 1–26.

SOLIMAN, M. A. AND ILYAS, I. F. 2009. Ranking with uncertain scores. In *Proceedings of the IEEE 25th International Conference on Data Engineering*. ICDE '09. IEEE Computer Society, Washington, DC, USA, 317–328.

SRIKANT, R. AND AGRAWAL, R. 1996. Mining sequential patterns: Generalizations and performance improvements. In *International Conference on Extending Database Technology*. EDBT '96. ACM, New York, NY, USA, 3–17.

TANAY, A., SHARAN, R., AND SHAMIR, R. 2002. Discovering statistically significant biclusters in gene expression data. *Bioinformatics 18*, 136–144.

TRAPP, A. C. AND PROKOPYEV, O. A. 2010. Solving the order-preserving submatrix problem via integer programming. *INFORMS J. on Computing 22*, 387–400.

WANG, H., NIE, F., HUANG, H., AND DING, C. 2011. Nonnegative matrix tri-factorization based high-order co-clustering and its fast implementation. In *Proceedings of the 11th IEEE International Conference on Data Mining*. ICDM '11. IEEE Computer Society, Washington, DC, USA, 774–783.

WELBOURNE, E., KHOUSSAINOVA, N., LETCHNER, J., LI, Y., BALAZINSKA, M., BORRIELLO, G., AND SUCIU, D. 2008. Cascadia: A system for specifying, detecting, and managing RFID events. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*. MobiSys '08. ACM, New York, NY, USA, 281–294.

YEUNG, K. Y., MEDVEDOVIC, M., AND BUMGARNER, R. 2003. Clustering gene-expression data with repeated measurements. *Genome Biology 4,* 5.

YIP, K. Y., KAO, B., ZHU, X., CHUI, C. K., LEE, S. D., AND CHEUNG, D. W. 2013. Mining order-preserving submatrices from data with repeated measurements. *IEEE Transactions on Knowledge and Data Engineering 25,* 7, 1587–1600.

ZHANG, M., WANG, W., AND LIU, J. 2008. Mining approximate order preserving clusters in the presence of noise. In *Proceedings of the IEEE 24th International Conference on Data Engineering*. ICDE '08. IEEE Computer Society, Washington, DC, USA, 160–168.

ZHAO, Z., YAN, D., AND NG, W. 2012. Mining probabilistically frequent sequential patterns in uncertain databases. In *Proceedings of the 15th International Conference on Extending Database Technology*. EDBT '12. ACM, New York, NY, USA, 74–85.

Received February xxxx; revised March xxxx; accepted June xxxx