Neurocomputing 336 (2019) 27–35

Contents lists available at ScienceDirect

Neurocomputing

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

Mining concise patterns on graph-connected itemsets

Di Zhang[a,*], Yunquan Zhang[a,b], Qiang Niu[c], Xingbao Qiu[d]

[a] School of Computer Science, Communication University of China, Beijing 100024, PR China
[b] State Key Lab of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, PR China
[c] Department of Mathematical Sciences, Xian Jiaotong-Liverpool University, Suzhou 215123, PR China
[d] China Mobile Communications Corporation, Beijing 100032, PR China

ARTICLE INFO

Article history:
Received 28 July 2017
Revised 21 February 2018
Accepted 21 March 2018
Available online 2 November 2018

Keywords:
Pattern mining
MDL
Graph
Diffusion kernel
Maximal entropy random walk

ABSTRACT

The itemset is a basic and usual form of data. People can obtain new insights into their business by discovering its implicit regularities through pattern mining. In some real applications, e.g., network alarm association, the itemsets usually have the following two characteristics: (1) the observed samples come from different entities, with inherent structural relationships implied in their static properties; (2) the samples are scarce, which may lead to incomplete pattern extraction. This paper considers how to efficiently find a concise set of patterns on such kind of data. Firstly, we use a graph to express the entities and their interconnections and propagate every sample to every node with a weight, determined by the pre-defined combination of kernel functions based on the similarities of the nodes and patterns. Next, the weight values can be naturally imported into the MDL-based filtering process and bring a differentiated pattern set for each node. Experiments show that the solution can outperform the global solution (trading all nodes as one) and isolated solution (removing all edges) on simulated and real data, and its effectiveness and scalability can be further verified in the application of large-scale network operation and maintenance.

1. Introduction

1.1. Pattern mining

Pattern mining aims to discover potential co-occurrence relationships among the items in a database. Classical frequent pattern methods, such as Apriori [1] and FP-Growth [2], extract qualified patterns by generating, sorting and filtering the possible candidates from data by merely counting their occurrences. These patterns can be used either as a final result for human analyzers or as intermediate features for subsequent data mining tasks, such as classification, clustering, etc. These methods are employed in wide variety of domains, for its intuitive design and ability to run fast.

However, some common defects exposed cannot be overlooked in practice. The main problem is the pattern explosion [3]. If a highly frequent sub-itemset passes the threshold examination, it will probably have many similar companions that also satisfy the test. In this case, it is likely that a large number of redundant results will be obtained. For human labors, it is tedious to check and comprehend them one by one; for data mining tasks, it may make the subsequent model overfit on latent noise and deteriorate its predicting accuracy. A straightforward approach here is to set the filtering threshold (i.e., support) high enough, to control the total number of patterns in a reasonable range. Nevertheless, this may cause the final results less informative, for they are so apparent that they sometimes can even be found only by bare eyes.

To solve this problem, people have switched their attention from frequent patterns to interesting or useful patterns. The critical issue is to re-design a more meaningful, and also computable optimization target. One popular approach is to filter out redundant patterns based on the MDL criterion, the Krimp algorithm [3]. It assumes that the set of patterns to be sought is a dictionary to encode the data, including code table itself and data body, and the corresponding compressed size is calculated based the total empirical entropy. Be aware of searching for the best combinations of patterns is an NP-hard problem, Krimp employs a heuristic approach to find a sub-optimal solution in polynomial time. Experiments in paper [3] show that, Krimp can generally reduce the number of outputs at least 2–3 orders of magnitude, and thus uncover those rare but helpful patterns.

* Corresponding author.
E-mail addresses: di.zhang@cuc.edu.cn (D. Zhang), zyq@ict.ac.cn (Y. Zhang), qiang.niu@xjtlu.edu.cn (Q. Niu), qiuxingbao@sn.chinamobile.com (X. Qiu).

https://doi.org/10.1016/j.neucom.2018.03.084
0925-2312/

1.2. Multi-tasking

Here we mainly consider how to apply the Krimp on itemsets with their structural relationships. In relational databases, the historical records are usually not produced namelessly, but with an identity field, marking who has generated them. Besides, there is typically a property table, where the identities act as a primary key instead of the foreign key in records, describing the static attributes of each entity, such as a user's basic information, or a networking device's uplink and downlink, etc. These properties makes sense in two respects. Firstly, it is possible to make some "reuse" of data between similar objects to improve the completeness and robustness of results, by analyzing the similarity between entities when the sample size is insufficient. Secondly, sometimes users are more concerned about each entity's specific patterns to see whether it has a unique personality, in contrast to the traditional global approach.

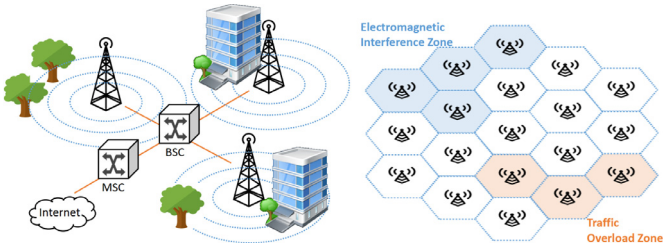
Similar to the scenario of multi-task learning, the key question here is how to acquire and exploit the relatedness of multiple tasks [4]. The cross-task structure can be either learned through the data or defined from prior knowledge. Once we have it, the relationship can be used to direct the data sharing between multiple tasks; or to provide a regularization for co-training multiple models; or, alternatively, to control a multi-output models complexity – in fact, all these three statements are equivalent. For our problem discussed here, unfortunately only a few samples can be collected, compared to the number of entities, and there is definitely no label in its unsupervised setting, thus we have to rely on domain knowledge to define the relevance and reflect it into the model through sensible regularization conditions.

The Krimp applies the MDL principle directly to the raw data through the self-defined heuristic search, which indeed bypasses the usual numerical optimization methods. A more practical multi-tasking solution is not to construct multiple coding tables simultaneously by some sophisticated tactics, but to make records under every node fully visible to each other, yet differentiate them by generating distinct weight matrices based on their relative positions and content. This similarity of nodes on a graph can be derived either by a classic random walk or a max entropy random walk. Next, the weights can be introduced directly into the Krimps criterion of evaluating pattern sets based on MDL. Moreover, this whole computation process is easy to be parallelized, and the performance can be enhanced almost linearly on a multi-core machine without much effort.

In the sections below, we first introduce the scenario of network alarm analysis and point out why the multi-task pattern mining is important, then summary the related works we have surveyed in Section 3. Next, some necessary background of compressing patterns, distribution embedding, and graph kernels are briefly given. In Sections 5 and 6, the theoretical model and algorithm design are presented in detail. A set of experiments are designed and conducted in Section 7. Finally, we summarize the whole article.

2. Motivated application: network alarm correlation

Nowadays, mobile communication has become a pivot ingredient for everyone's life. Wireless carriers, to provide such a service, deploy a grid of base stations at an appropriate density to achieve ubiquitous coverage of a geographic area and provide adequate bandwidth with reliability in a load-balanced manner. This type of network is often referred to as a cellular network because its stations are laid out in a hive-like layout (Fig. 1(b)), though the coverage area of one tower is of course not strictly hexagonal due to geographical factors. They are usually massively deployed, up to tens of thousands for one telecom site, to cover a densely populated metropolis or large tracts of rural areas in a much sparser way.



(a) Typical Structure of Network (b) Layout of Base Stations

Fig. 1. Wireless network architecture.

Cellular networks, typically designed as hierarchical structures (Fig. 1(a)): those adjacent base stations are grouped and connected to one BSC (Base Station Controller), and dozens of BSCs are connected to one MSC (Mobile Switching Center) at upper layer, and MSCs further link up to the upper layer, and so on. When a phone uploads its data packets, the traffic flow travels bottom-up via a layer-by-layer aggregation to reach service providers on the Internet; conversely, the streams go top-down in the round trip. Thus we can understand those nodes that are adjacent, or those nodes linked up to one common ancestry may exhibit similar phenomenon to the network administrator. For example (Fig. 1(b)), if there is a periodical electromagnetic interference caused by a possible subway construction nearby, the stations in the same district will be all affected. Another example, sometimes one BSC happened to have its hardware resource exhausted, then the devices linked to it may experience bandwidth drop-down at the same time.

Since the real cause of a network failure is unobservable, the administrator can only deduce it by analyzing the alarm logs from the devices and check if they exactly match a particular pattern appeared before. For instance, when a device reports a high rate of re-transmission and a sharp drop-down in access success rate, there may be an overloading of the equipment caused by a traffic burst in the neighboring area. One critical issue concerned by the administrator is that how to exploiting commonalities and differences across nodes. The alarms may be fakely activated or improperly screened by some fault tolerance policy. A specific external or internal problem does not necessarily lead to all the base stations alerting, but only a small part of the devices sporadically will report up. For the operator, it is hard to conduct a comprehensive analysis on historical records, neither from a separated perspective of each equipment nor viewing them as all the same. The network topology must be concerned, to merge these initially scattered records in a justified way.

3. Related work

3.1. Structural pattern mining

A recent survey of alarm correlation methods can be found in [5]. There are two approaches for this issue: the rule-based and data-based, the former mainly relies on experts to define hard-coded rules, while a large part of the latter preferred the pattern mining to generate rules on the machine itself. The initial work on mining association rules on alarms can be found in [6], which built a semi-automatical system in service mainly based on rules from human, and let the machine propose candidate rules to experts and leave the right of confirming to them. A similar but much newer data-based rule discovery system can be found in [7]. Given the variety of alarm sources, work [8] can output multi-sequential

patterns aligned by their timestamps. Paper [9] considered how to only aggregate those nearby alarms, by imposing a constraint based on network structure. After extensive searching, we did not find the same problem discussed here in the published literature.

Meanwhile, a class of closely related topics has drawn attention to the researchers in related fields. In the social networking applications, the paper [10] uses a hypergraph to describe the interactions between users and the commodities, with its nodes representing the purchasing behavior and hyperedges describing the two-way effects between users and the products and designs a greedy algorithm to select a node seed collection for the recommendation on products. In the paper [11], a method of subgraph pattern mining based on sampling techniques is introduced to study how users interact with each other in social networks. In the field of graph mining, the paper [12] and [13] consider how to detect the frequent dense subgraphs, all nodes of which share at least one common attributes. In the paper [14], the problem was extended to consider topological factors (connectivity, centrality) besides attributes for each node. The co-evolving patterns between structure and attributes in dynamic graphs are further studied in paper [15]. In short, their problem definitions and employed techniques are quite different from ours, mainly on what we treat the records on nodes as repeatable observations, instead of a group of fixed attributes.

3.2. Kernel methods

There are two themes to be introduced, how to embed the distribution into reproducing kernel Hilbert space (RKHS) [16], and how to define similarity for nodes on one graph. The skills of embedding the distributions into infinite-dimensional feature spaces, as a generalization of the individual data-point feature mapping done in classical kernel methods, allows one to manipulate distributions using Hilbert space operations such as inner products, distances, projections, and spectral analysis [17]. The method has already applied to many applications, such as measuring distance between distributions, density estimation, measuring dependence, and distribution regression [18]. For the multi-tasking, paper [4] gives a survey on kernels for vector-valued functions.

For nodes on a graph, a common kernel function is the diffusion kernel [19]. The article [19] illustrates the hypothesis from a random walk, from which the diffusion kernel can be derived. Its applications include label propagation [20], community detection [21]. For the scenario of demands for more locality in traveling, a popular alternative is the maximal entropy random walk [22], which has been applied for link prediction [23], community detection [24] and centrality measures [25]. The maximum entropy random walk (MERW), which is based on the global centrality of nodes, can also be generated based on local information in paper [26].

3.3. MDL for pattern mining

The widely used algorithms of itemset pattern mining include Aproiri [1] and FP-growth [2]. To solve the problem of pattern redundancy, the early methods include closed patterns [27], interesting patterns [28]. Paper [29] claims that the MDL is always a remarkable approach to data mining. Krimp [3] is a breakthrough effort to exploit the MDL guidelines for filtering redundant itemset patterns. The method can be further applied to sequence [30], graph [31], and stream [32], and can be used for privacy preservation [33], missing value estimation [34], and change detection [35]. It is worth noting that the paper [35] constructs multiple coding tables for anomaly detection. In respect to performance, the paper [36] puts forward a method of mining the model directly on the raw data, and the paper [37] has parallelized Krimp. Also, the related works to information theory include, the paper [38] discussed the entropy measure of a graph, and the paper [39] mentioned the entropy calculation on weighted samples.

4. Preliminaries

4.1. MDL-based pattern mining

Let $\mathcal{A}$ be a character set, and an itemset I be a non-empty subset of $\mathcal{A}$. A database $\mathcal{D} = \{t_1, \ldots, t_n\}$ is a collection of transactions, and each transaction t is an itemset. The pattern X is also an itemset, which may appear in multiple transactions. Usually, we say that the transaction t supports X, if and only if $X \subseteq t$. Obviously, all the subsets of a transaction support it. The set of patterns contained by a database $\mathcal{D}$ is $\bigcup_{t\in\mathcal{D}}\{X|X \subseteq t\}$.

For frequent pattern mining, the problem is to find out all the patterns whose frequency, or support $sup(x) = |\{t \in \mathcal{D}|X \in t\}|/|\mathcal{D}|$ is greater than a pre-defined threshold ϵ. For compressing mining, the set of patterns to be found are instead to be evaluated by its ability to encode the data and itself compactly. The workable sets are of course not unique. For instance, given $\mathcal{D} = \{\{A, B, C\}, \{A, C\}\}$, both $CT_1 = \{\{A, C\}, \{B\}\}$, a 2-pattern set, and $CT_2 = \{\{A\}, \{B\}, \{C\}\}$, length of 3, can encode it. In the Krimp algorithm, the MDL criterion in the information theory is used to judge the merits of the pattern set, that is, to compress the data as small as possible on the one hand, and to prevent the code table itself from being too large. The total compressed length of database D is (i.e., the optimization target) [3]

$$L(\mathcal{D}, CT) = L(CT) + L(\mathcal{D}|CT). \tag{1}$$

The latter represents the length of the data encoded. Specifically, we have

$$L(\mathcal{D}|CT) = \sum_{t\in\mathcal{D}} L(t|CT), \tag{2}$$

$$L(t|CT) = \sum_{X\in Cover(CT,t)} L(code_{CT}(X)), \tag{3}$$

where *Cover* function gives the matching words $code_{CT}$ for transaction t based on CT. In addition, the length of the code table is

$$L(CT) = \sum_{X\in CT:Usage_{\mathcal{D}}(X)\neq 0} L(code_{ST}(X)) + L(code_{CT}(X)), \tag{4}$$

where the former item represents the entries in the coding table and the latter represents the corresponding code length, and $Usage_{\mathcal{D}}(X)$ gives the number of occurrences of each pattern X.

Solving an optimal coding table is an NP-hard problem [3]. In general, a heuristic search is used to obtain a feasible solution. In the search process, we need to define two orders: (1) the coding order specified by the Cover function in Eq. (3), that is, the matching priority of the entry for the data, which uniquely determines the encoding of a transaction; (2) the order of the candidate patterns to be added to the coding table to assess whether it can further enhance the compression effect [36]. The order of candidates is mainly based on the descending sorting of Usages results, i.e., high-frequency patterns first. We will try to change the Usage in step (2) later.

4.2. Distribution embedding and graph kernels

Let X denote a random variable with codomain Ω and distribution $P(X)$. In the presence of a kernel κ on $\Omega \times \Omega$, there must be an RKHS $\mathcal{H}$ satisfying the reproducing property. Alternatively, $\kappa(x, \cdot)$ defines a feature mapping $\phi(x)$ from Ω to $\mathcal{H}$, so that $\kappa(x, x') =< \phi(x), \phi(x') >_{\mathcal{H}}$ can be viewed as a measure of similarity between point $x, x' \in \Omega$. The Ω can be composed of commonly

seen vectors, or more powerfully, be any objects we can define κ or $\phi(x)$. We can embed a probability distribution function into the above space as the way of 'mean map':

$$\mu_X := \mathbb{E}_X[\kappa(x,\cdot)] = \mathbb{E}_X[\phi(x)] = \int_\Omega \phi(x)dP(x). \tag{5}$$

Given n samples $\{x_1,\ldots,x_n\}$, the Definition 5 can be empirically written as:

$$\hat{\mu}_X = \frac{1}{n}\sum_{i=1}^{n}\phi(x_i). \tag{6}$$

Correspondingly, the operators of the joint probability and conditional probability can be defined as:

$$\mathcal{C}_{XY} := \mathbb{E}_{XY}[\phi(X)\otimes\phi(Y)] = \int_{\Omega\times\Omega}\phi(x)\otimes\phi(y)dP(x,y), \tag{7}$$

$$\mathcal{C}_{Y|X} := \mathcal{C}_{YX}\mathcal{C}_{XX}^{-1}, \tag{8}$$

where $\otimes$ is the tensor product. Furthermore, we can define kernel functions for the nodes of graphs. An undirected graph G is defined by a vertex set V and a edge set E, and E is a set of disordered pairs. Define adjacency matrix $\mathbf{A}:=[a_{ij}]_{m\times n}$, where $a_{ij}=1$ if $\{v_i,\ v_j\}\in V$, else $a_{ij}=0$; Capacity matrix $\mathbf{D}:=diag([d_i]_{n\times 1})$, where $d_i=\sum_{v_i\in V}a_{ij}$; Laplacian matrix $\mathbf{L}:=\mathbf{D}-\mathbf{A}$ and its random walk normalized variety is $\mathbf{L}_r:=\mathbf{D}^{-1}\mathbf{L}=\mathbf{I}-\mathbf{D}^{-1}\mathbf{A}$ [23]. Then the diffusion kernel is defined as:

$$\kappa_G := e^{\theta\mathbf{L}_r} = \lim_{s\to\infty}\left(\mathbf{I}+\frac{\theta\mathbf{L}_r}{s}\right)^s \quad s\in\mathbb{N}, \tag{9}$$

where θ denotes the bandwidth parameter for adjusting the strength of diffusion. Definition 9 can be derived from the differential Eq. (10) describing the evolution of states while random walk happens on the graph:

$$\frac{\partial}{\partial\theta}e^{\theta\mathbf{L}_r} = \mathbf{L}_r e^{\theta\mathbf{L}_r}. \tag{10}$$

The ordinary random walk here is unbiased, its transition probability $\mathbf{S}$ for each node is

$$\mathbf{S}_{ij} = \mathbf{A}_{ij}\Big/\sum_k \mathbf{A}_{ik}. \tag{11}$$

In some cases, we hope that the nodes with a higher centrality on a graph can gain greater weights. In this scenario, we prefer to choose the MERW, whose transition probability is proportional to the dominant eigenvector $\boldsymbol{\psi}$ of the adjacency matrix, that is

$$\mathbf{S}_{ij} = \frac{\mathbf{S}_{ij}}{\lambda}\frac{\boldsymbol{\psi}_j}{\boldsymbol{\psi}_i}, \quad \max_{|\lambda|}\mathbf{S}\boldsymbol{\psi}=\lambda\boldsymbol{\psi}. \tag{12}$$

It can be shown that this walk can maximize the entropy rate $H(\mathbf{S})$ of the stochastic process to $\log\lambda$ [22].

5. Model

First of all, a global model is required to estimate the distribution density for all nodes with a limited amount of samples. The distribution density, in the form of weighted average of mapped existing samples, is estimated with two regularizations: one for preventing overfitting on noise, and the other for dumping the influence of every observed sample. Second, the available kernels, including graph and content, are combined in a separable form to enable the optimization process. A particular form of kernel derived from the MERW is provided to encourage a more localized measurement of node similarity. Finally, the importance of every sample for every node is incorporated into the MDL principle to filter out the concise result sets.

5.1. Density estimation

Assume that there is an underlying probability mass function: $P_T := P_{V,X} : |\mathcal{V}|\times\{0,1\}^{|\mathcal{A}|}\to[0,1]$, where T is the tuple of V and X, and V denotes for the indexing number ranging from 0 to $|\mathcal{V}|$ in integer, and X denotes an observation contained in all enumerations of characters of alphabet $|\mathcal{A}|$, including the null pattern $\{0,\ldots,0\}_{|A|}$. In information theory, after samples $\mathcal{D}=\{t_i\}_{i=1}^N=\{(v_i,\mathbf{x}_i)\}_{i=1}^N$ are drawn from distribution P_T, we assume that the probability to be estimated should minimize the following objective function:

$$KL(P_T||\hat{P}_T) := \int_\Omega \log\left(\frac{dP_T}{d\hat{P}_T}\right)dP_T, \tag{13}$$

where KL is the Kullback–Leibler divergence and $\hat{P}_T$ is the empirical distribution estimated from $\mathcal{D}$. In the information theory approach, the computation of KL-divergence must depend on either perform a parameter estimation for probabilities or employ a sophisticated space-partitioning/bias-correction strategies which typically infeasible for high-dimensional data [17]. In the kernel method, by mapping the distribution to the Hibert space, actually bypassing this step, and the distance between two points in the space is utilized to measure the difference between distributions:

$$D(P_T,\hat{P}_T) := \|\mu[P_T]-\mu[\hat{P}_T]\|_{\mathcal{H}}, \tag{14}$$

where $\|\cdot\|_{\mathcal{H}}$ is the norm of Hibert space. Directly setting the Definition 14 as the optimization goal will lead to two problems in practice. One is overfitting, which is to be suppressed by the well-known Tikhonov regularization. The other is the itemset specific, induced by the phenomenon that not all nodes have observable samples, especially in alarm applications: a burst of alarms in one node does not necessarily mean that all nodes, including those who kept silent all the time, will have a higher probability of alarming after that. Thus, the diffusion of observed records will be damped by an extra regularization term, defined by the distance between P_T and P_{T_s}, and those silent nodes will intend to recover to all-0 distribution gradually along the path of alarms' spreading. The revised optimization goal is:

$$\min_{P_T}\|\mu[P_T]-\mu[\hat{P}_T]\|_{\mathcal{H}}^2+\alpha\|\mu[P_T]\|_{\mathcal{H}}^2+\beta\|\mu[P_T]-\mu[P_{T_s}]\|_{\mathcal{H}}^2, \tag{15}$$

where $0\le\alpha\le 1$ and $0\le\beta\le 1$. On the other hand, the convex composition of samples can be used to approximate its underlying distribution [17]. We get the following new representations (compared to Eq. (6)), where the symmetric matrix $\mathbf{W}\in\mathbb{R}^{N\times N}$ contains all the weights we want:

$$\mu[P_T]\approx\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N}\mathbf{W}_{i,j}\phi(t_j), \tag{16}$$

where $\sum_{i=1}^N\mathbf{W}_{i,\cdot}=1$ and $\mathbf{W}_{i,j}\ge 0$. The $\mathbf{W}$ can be directly set to $\mathbf{I}$ if we need to use the empirical distribution without rectification. Substituting Eqs. (6) and (16) into Eq. (15), we get the transformed target:

$$\min_{\mathbf{W}}\|\mathbf{W}^T[(1+\beta)\mathbf{K}+\alpha\mathbf{I}]\mathbf{W}-\mathbf{W}^T(\mathbf{K}+\beta\mathbf{K}^s)-(\mathbf{K}+\beta\mathbf{K}^{sT})\mathbf{W}\|_{1,1}, \tag{17}$$

where $\mathbf{K}_{i,j}:=\kappa(t_i,\ t_j)$, $\mathbf{K}^s_{i,j}:=\kappa_I(\mathbf{x}_i,\{0,\ldots,0\}_{|\mathcal{A}|\times 1})$ for $t_i,t_j\in\mathcal{D},\forall i,j\in 1,\ldots,N$. The κ_I is a predefined kernel on the content of samples. The $\|\mathbf{A}\|_{1,1}$ is the sum of all entries in a matrix $\mathbf{A}\in\mathbf{R}^{N\times N}$, and it also can be written as $\|\mathbf{A}\|_{1,1}=\mathbf{1}^T\mathbf{A}\mathbf{1}$, given $\mathbf{1}:=[1,\ldots,1]_{N\times 1}$. Noticed that $\mathbf{W}^T=\mathbf{W}$, we can flatten the $\mathbf{W}$ into a $\frac{N(N+1)}{2}$−length of vector $\mathbf{w}^*$ by removing all duplicate entries, i.e., $\mathbf{W}_{j,i}=\mathbf{W}_{i,j}=\mathbf{w}^*_{N(i-1)-\frac{(i-1)(i-2)}{2}+j}$, $\forall i\ge j$, thus Eq. (17) can

be rewritten as:

$$\min_{\mathbf{w}^*} \mathbf{w}^{*T}\mathbf{K}^*\mathbf{w}^* - (\mathbf{K}^{**T} + \mathbf{K}^{***})\mathbf{w}^*, \tag{18}$$

where $\mathbf{K}^*$, $\mathbf{K}^{**}$ and $\mathbf{K}^{***}$ are the $\{1 : \frac{n(n+1)}{2}; 1 : \frac{n(n+1)}{2}\}$ sub-matrices of $N \times N$ repetitions of matrices $(1+\beta)\mathbf{K} + \alpha\mathbf{I}$, $\mathbf{K} + \beta\mathbf{K}^s$ and $\mathbf{K} + \beta\mathbf{K}^{sT}$ appeared in Eq. (17). So far, the problem has been completely transformed into a constrained quadratic programming problem. It can be solved by the classical SMO algorithm [40], which only deals with two variables at every iteration, without storing the whole Gram matrices, e.g., $\mathbf{K}^*$ in Eq. (18), in memory.

5.2. Kernel design

The kernel κ for two samples $t_i = (v_i, \mathbf{x}_i)$ and $t_j = (v_j, \mathbf{x}_j)$ is assumed to be the product of two simple kernels:

$$\kappa((v_i, \mathbf{x}_i), (v_j, \mathbf{x}_j)) = \kappa_G(v_i, v_j)\kappa_I(\mathbf{x}_i, \mathbf{x}_j), \tag{19}$$

where the κ_G represents the similarity of nodes on a graph, and the κ_I is the similarity of records only based on their binary content. These kernel functions can be designed based on the user's domain knowledge or subjective preference, and their significant differences will be observed in Section 7.

For κ_I in formula (19), we have two options: a straightforward choice is using the Dirac delta kernel [4], which means there is no similarity at all even if two samples are slightly different.

$$\kappa_I^{(1)}(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} 1, & \mathbf{x}_i = \mathbf{x}_j \\ 0, & \mathbf{x}_i \neq \mathbf{x}_j \end{cases}. \tag{20}$$

Another option is to consider the cosine kernel borrowed from document comparison in NLP [41].

$$\kappa_I^{(2)}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i^T\mathbf{x}_j}{\|\mathbf{x}_i\|_2\|\mathbf{x}_j\|_2}. \tag{21}$$

where $\|\cdot\|_2$ is the L2-norm for finite-length vectors. For the κ_G, we have two choices. One is the diffusion kernel linked to the ordinary random walk, $\kappa_G^{(1)} = e^{\theta\mathbf{L}_r}$. Another one is the MERW kernel, designed for emphasizing the locality of nodes, especially for the tree-like structure of a wireless network. From the changed transition probability in the definition of Eq. (12), it is equivalent to redefine the adjacency matrix as [23]

$$\mathbf{A}'_{ij} := \gamma_i\boldsymbol{\psi}_i\boldsymbol{\psi}_j, \tag{22}$$

because the same Eq. (12) can be derived from Eq. (22): $\mathbf{S}_{ij} = \frac{\mathbf{A}_{ij}\gamma_i\boldsymbol{\psi}_i\boldsymbol{\psi}_j}{\sum_j \gamma_i\boldsymbol{\psi}_i\boldsymbol{\psi}_j} = \frac{\mathbf{A}_{ij}\boldsymbol{\psi}_j}{\lambda\boldsymbol{\psi}_i}$. The γ is a parameter vector setting a scaling factor for each row of $\mathbf{A}'$. To keep the capacity matrix $\mathbf{D}$ consistent and make θ in Eq. (9) comparable in different graph kernels, we set

$$\gamma_i = \mathbf{D}_{ii} = \sum_{v_i \in V} a_{ij}. \tag{23}$$

Then we have the new Laplacian and kernel definition, following Eq. (9):

$$\mathbf{L}_m := \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}', \tag{24}$$

$$\kappa_G^{(2)} = e^{\theta\mathbf{L}_m}. \tag{25}$$

6. Implementation

6.1. Procedure

Let us analyze the complexity of the Algorithm 1 step by step. The step 4 needs $\mathcal{O}(|V|^3)$ time to complete, and the step 5 needs only $\mathcal{O}(|E|) \leq \mathcal{O}(|V|^2)$ and can be omitted later; The line 7 calls a matrix exponential function as the name of scipy.linalg.expm routine [42], using the Pade approximation [43] which complexity can be rough estimated as $\mathcal{O}(|V|^3)$ [44]; the SMO method in step 10 requires $\mathcal{O}(|\mathcal{D}|^{2.2})$ empirically to converge [40]; we can summarize the running time of all the above steps to $\mathcal{O}(|V|^3 + |\mathcal{D}|^{2.2})$. For each node, the cost of Krimp in step 13 is $\mathcal{O}(|CT|\log|CT| + |\mathcal{D}| \times |CT| \times |\mathcal{A}|)$, then the overall complexity of the whole procedure is $\mathcal{O}(|V| * (|CT|\log|CT| + |\mathcal{D}| \times |CT| \times |\mathcal{A}|) + |V|^3 + |\mathcal{D}|^{2.2})$. For the usual situation, we have $|V| < |\mathcal{D}|$, thus the impact of steps 3–10 on performance is secondary compared to the later steps 12–14.

Algorithm 1: The main procedure of our solution.

Data: A database $\mathcal{D}$, a graph G, a configuration ($type\_of\_\kappa_I$, $type\_of\_\kappa_G$, α, β)
Result: A pattern set CT

```
1  // define kernels
2  kernel function κ_I ← if * then type_of_κ_I=="Dirac";
3  κ_I^(1) else *;
4  κ_I^(2) ;
5  if type_of_κ_G=="MERW" then
6  |  Dominant eigenvector ψ ← eigen_decomposition(A) ;
7  |  A ← redistribute weight of A ;
8  end
9  Gram matrix K_G ← generate similarity κ_G = e^{θL_r} for all nodes;
10 // compute weights
11 kernel function κ ← κ_I * precompute(K_G);
12 weight matrix w ← SMO(optimization taget, κ);
13 // pattern mining
14 foreach v_i ∈ V do
15 |  CT_i ← krimp(D, w_i);
16 end
17 return CT ← ⋃ CT_i
```

6.2. Parallelization

Though the loop between lines 12–14 is time-consuming, it can be easily parallelized on any multi-core machine, since all computation process on nodes are indeed independent to each other, without any communication interactions or exclusive policy on memory accessing once these sub-tasks are dispatched. The running time will hopefully gain a nearly linear scalability as the resource increases, which means given p processors, and the running time will drop-down to $\mathcal{O}(|V| \times \text{time(Krimp)}/p + \text{time(preprocessing)})$.

7. Experiments

7.1. Data

We use two types of data to verify the effectiveness of the solution, including the synthesis and real dataset based on the scenario in Section 2. There are two main reasons for generating a simulated dataset: (1) whether the behavior of the algorithm is consistent with what we expect, and the key properties of the dataset, such as the region of the items occurrence, the specific characters it contains, etc., can be freely controlled as needed. (2) Because the real data, or even possible clues to deduce the source of data, are not allowed to be published according to the commercial protocol, we need to reproduce a pastiche of the data based on the main characteristic of our scene, such as geographically distri-

Table 1
Simulation dataset descriptions.

Name	Position	Character	#Non-empty nodes	#Non-empty chars
sim 1	Random	Random	650	10
sim 2	Upper right area ($lon > 0$, $lat > 0$)	Random	249	10
sim 3	Random	Small ($a \in [0, 4]$)	650	5
sim 4	Upper right area	Small	249	5

Table 2
Alarm dataset descriptions.

Name	Vertex	Edge	Record	$\|V\|$	$\|E\|$	$\|A\|$
net1	Cell	Connection	Alarm	11,425	16,508	102
net2	Base station	Connection	Alarm	21,575	32,659	85
net3	Base station	Connection	Alarm	8462	10,481	38

bution, tree-like connection, to demonstrate the impact of various settings on the final results.

Simulated data. Firstly, we use NetworkX [45] to generate random graphs, whose nodes ($|V| = 1000$) can be randomly placed on a two-dimensional plane (a rectangle between (0,0) and (1,1)), and a tree topology produced by the minimum spanning tree algorithm. This is similar to the shortest connection principle in the network design. Secondly, we independently generate a small but fixed number of binary records ($|A| = 10$, $|\mathcal{D}| = 1000$) for each node, whose locations of the 1 s can be random, or concentrated in a subset of $\mathcal{A}$. To make the tests comprehensive, we designed four sets of data, combining the globality, locality of the geographical, alphabetic factors, as shown in Table 1.

We have two steps for generating samples: (1) distribution definition. The occurrences of every character of every node all conform to the Bernoulli distribution $Ber(0.5)$, independently. The marginal probability of the position where the next record appears is also set to be exactly equal, that is, to obey the categorical distribution $Cat((1/|V|)^*|V|)$. 2) sampling. A specified number of samples are generated from the defined distribution (excluding all-0s), and formatted as itemsets for later use. To avoid sophisticated assumptions, we do not intend to propose any correlation between characters in the simulation and check patterns based on that, but resort to the real data for such kind of validation.

Network alarm data. We collected the alarm records from three different cellular networks in different regions, as shown in the Table 2.

7.2. Design

We plan to check the effectiveness of the whole scheme at two levels: one is the accuracy of the probability density estimation for each node, as the intermediate result; the other is the validity of the obtained pattern, as the final result. For the former, we use mean integrated squared error $MISE(f, \hat{f}) = \frac{1}{n}\sum_i^n (f(x_i) - \hat{f}(x_i))^2$ to evaluate the average error between the estimated value and the ground truth, which is only available for the simulated data. Limited by the sample size, the real data can merely evaluate the pattern minings criteria: the first is the actual size the code table, having removed those trivial singleton code elements, to check whether we get target patterns; the second one is to see if the data have been really compressed with these patterns, compared to the ordinary singleton code table. These two aspects are quantified by the number of non-singleton patterns $|CT/\mathcal{A}|$ and the compression ratio $\frac{L(D,CT)}{L(D,ST)} \times 100\%$ respectively.

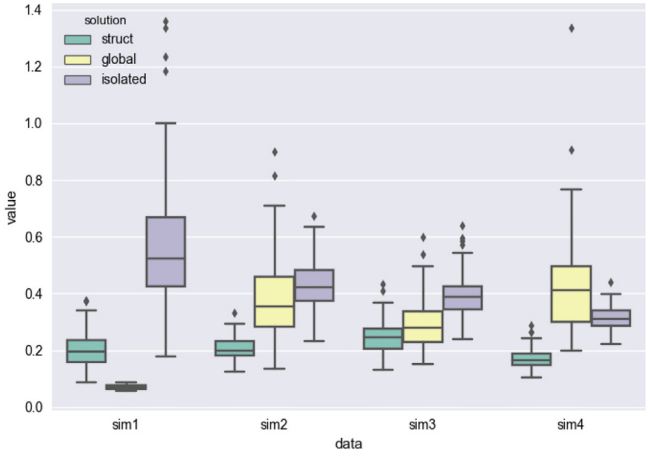


Fig. 2. Result of density estimation on simulated data.

Table 3
Optimal settings for datasets.

Name	Graph kernel	Item kernel	α	β
sim1	All-one	Cosine	10^3	10^0
sim2	MERW	Cosine	10^1	10^{-1}
sim3	All-one	Dirac	10^1	10^{-1}
sim4	MERW	Dirac	10^0	10^{-1}
net1	ORW	Dirac	10^0	10^{-1}
net2	MERW	Dirac	10^1	10^{-1}
net3	MERW	Dirac	10^1	10^{-1}

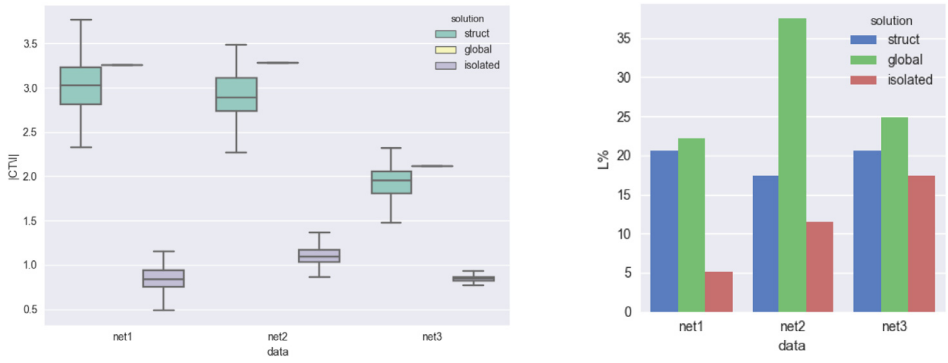
Two additional solutions are implemented for comparison, the global mining and the isolated mining. The global mining means simply mixing all samples at all nodes, ignoring the structure information, and compress them as a whole. This is equivalent to use the kernel $Constant(1)^*Dirac$, while the isolated one uses the kernel $Dirac^*Dirac$, make each nodes data invisible to others. Both of them belong to the traditional approach suitable for acting as benchmarks. Furthermore, the parameters we can set contains regularization parameter α, β and the choice of kernels(the bandwidth parameter θ of diffusion kernels can be set by default as 1, for it has the similar impact on results to α in practice). Their impact on accuracy will be confirmed in experiments, and the best configurations will be found for each dataset. Finally, we will test the running time reduction from parallel acceleration. It is evaluated by the speedup ratio $S(p) = \frac{p}{\beta p + (1-\beta)}$, where β is the sequential part of running time. Particularly, if $p \to \infty$, then the upper bound of $S(p)$ is $S_m = 1/\beta$; if $\beta = 0$, $S_m \to \infty$, then a perfect linear speedup will be reached in theory.

7.3. Result

7.3.1. Comparison of solutions

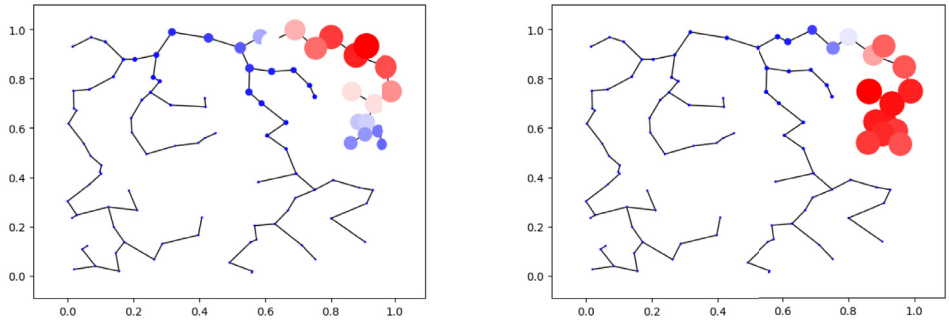
For the sim1 data sets, it is most suitable to directly apply the global mining solution, since its generation process does not take into account any structured setting and make itself have same and independent distribution. When the limitation is set according to the graph or alphabetic neighborhood relationship in sim2 and sim3, the structural mining starts to outperform the traditional ones, and achieve obviously the best result by setting proper kernels at sim4. In all, the experimental result in Fig. 2 on simulated data verify the feasibility of the solution and confirm it behaves like what we expected.

In Fig. 3, the overall validity of the result are studied from both non-trivial code symbols (size of the model) and length of compressed records (size of the data). When the model is too large, it is prone to overfit; whereas the model is too simple, the potential



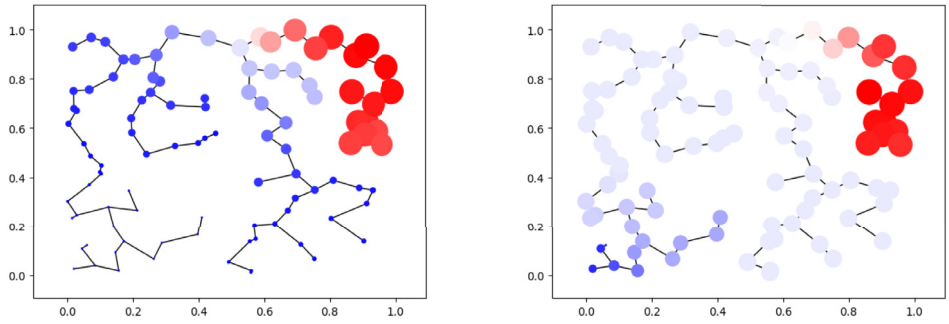
(a) Number of patterns (b) Compression Ratios

Fig. 3. The quality of concise patterns.



(a) κ_G=ORW, κ_I=Dirac, $\alpha = 10^1$, $\beta = 10^0$

(b) κ_G=MERW, κ_I=Dirac, $\alpha = 10^1$, $\beta = 10^0$



(c) κ_G=MERW, κ_I=Dirac, $\alpha = 10^2$, $\beta = 10^0$

(d) κ_G=MERW, κ_I=Dirac, $\alpha = 10^1$, $\beta = 10^{-3}$

Fig. 4. Effect of configurations. The size and color of nodes in the tree are both computed by its $\text{argmax}_{x \in A} P(x)$. The redder and bigger nodes mean a larger value, and vice versa.

correlation in the data is not fully exploited. We turn to real data to study this phenomenon, given the occurrence of characters are independent of each other. In the network alarms, each node personalizes its code table for their special samples, and may achieve better compression ratio with comparable, or even less extracted patterns.

7.3.2. Impact of hyper-parameters

The adjustable hyper-parameters in the optimization target (17) and (19) includes, κ_G, κ_I, α and β, which may vary for different dataset shown in Table 3. We employ a global optimization process, e.g., grid search, on the combination of all possible values, constructed by adding the All-one and Dirac kernel into the candidate set, and discretizing the scope of α, β to the set of $10^{\{-3,-2,-1,0,1,2,3\}}$. The configuration on simulation data coincides with the assumption we have made in Section 7.1, as the choices of kernels, keeps matching the various limitations imposed in the generation, and the regularization also becomes looser as the samples become denser. The result on real data also reveals part of their underlying properties. The most obvious difference is the graph kernel. Since the net1s alarms are collected from the cells, at a subordinate level from base stations, where the same number of units may cover a relatively smaller area, they may not be able to exhibit enough locality than larger networks, such as the net2 and net3. Other settings for network data are quite similar, only with a slight difference in regularization parameters.

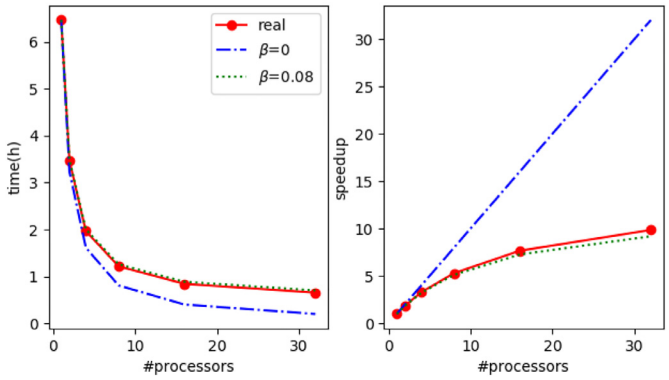


Fig. 5. Speedup by parallelization. The net2 dataset is selected for test, for its has the largest number of nodes $|V| = 21575$.

The influence of hyper-parameters can be further explored from the visualization of simulated data. Fig. 4(a) gives a standard diffusion starting from the upper right corner of the plane with only one active character, where the color and size of nodes both display the intensity of the most frequent symbol. The strength of the symbol gradually damped to zero when the positions go leftward and downward, while Fig. 4(b) depicts what will happen if we replace the ORW with the MERW. It is easy to observe that the locality emerges and the propagation along the tree are suppressed. Fig. 4(c) tells us the estimation will be enforced to consider more global information if we set the alpha to a larger value and Fig. 4(d) shows that the absence of a large enough β will result in those silent nodes resonates unnecessarily with those beeping ones.

7.3.3. Parallelization

The parallelized program ran on a 32-core server in a multi-thread way, one thread per node, and achieved a sub-linear speedup while an increasing number of cores are involved, shown in Fig. 5. Close to the real speedup points, for comparison, we draw a theoretic curve with $\beta = 0.08$, which grantees a potentially high S_m, i.e., good scalability on hardware resource.

8. Conclusion

In this paper, we have implemented and tested a solution built upon a two-phase framework: (1) a kernel-base multi-task density estimation, representing each target probability as a combination of all existing samples blending by differentiated weights, derived from kernels based on business understanding; (2) these mutually compensated samples, can be easily imported into the entropy calculation of Krimp algorithm. This solution builds a bridge between structural collaboration in multi-tasking and compression-based data mining technologies. The application of network alarm correlation has validated its effectiveness, customizability, and ease of implementation.

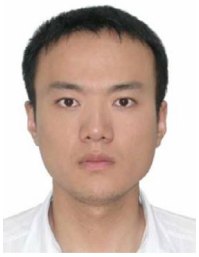
Acknowledgments

The work is partially supported by NSF of China under no. 11301420; NSF of Jiangsu Province under nos. BK20150373 and BK20171237; Suzhou Science and Technology Program under no. SZS201613 and the XJTLU Key Programme Special Fund (KSF) under no. KSF-A-01.

References

[1] R. Agrawal, R. Srikant, et al., Fast algorithms for mining association rules, in: Proceedings of the Twentieth International Conference on very large data bases (VLDB), 1215, 1994, pp. 487–499.
[2] J. Han, J. Pei, Y. Yin, Mining frequent patterns without candidate generation, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, ACM sigmod record, 29, ACM, 2000, pp. 1–12.
[3] J. Vreeken, M. Van Leeuwen, A. Siebes, Krimp: mining itemsets that compress, Data Min. Knowl. Discov. 23 (1) (2011) 169–214.
[4] F. Dinuzzo, Learning output kernels for multi-task problems, Neurocomputing 118 (2013) 119–126.
[5] S.A. Mirheidari, S. Arshad, R. Jalili, Alert correlation algorithms: a survey and taxonomy, in: Cyberspace Safety and Security, Springer, 2013, pp. 183–197.
[6] M. Klemettinen, H. Mannila, H. Toivonen, Rule discovery in telecommunication alarm data, J. Netw. Syst. Manag. 7 (4) (1999) 395–423.
[7] J. Wang, C. He, Y. Liu, G. Tian, I. Peng, J. Xing, X. Ruan, H. Xie, F.L. Wang, Efficient alarm behavior analytics for telecom networks, Inf. Sci. 402 (2017) 1–14.
[8] S. Lai, T. Chen, A method for pattern mining in multiple alarm flood sequences, Chem. Eng. Res. Des. 117 (2017) 831–839.
[9] Z.-d. Zhao, H. Nan, Z.-h. Li, Alarm correlation analysis in SDH network failure, in: Proceedings of the National Conference on Information Technology and Computer Science, Atlantis Press, 2012.
[10] H.-J. Hung, H.-H. Shuai, D.-N. Yang, L.-H. Huang, W.-C. Lee, J. Pei, M.-S. Chen, When social influence meets item inference, in: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, ACM, 2016, pp. 915–924.
[11] J. Zhang, J. Tang, Y. Zhong, Y. Mo, J. Li, G. Song, W. Hall, J. Sun, StructInf: mining structural influence from social streams, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, 2017, pp. 73–80.
[12] A. Anagnostopoulos, R. Kumar, M. Mahdian, Influence and correlation in social networks, in: Proceedings of the Fourteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2008, pp. 7–15.
[13] A. Silva, W. Meira Jr, M.J. Zaki, Structural correlation pattern mining for large graphs, in: Proceedings of the Eighth Workshop on Mining and Learning with Graphs, ACM, 2010, pp. 119–126.
[14] J.-F. Boulicaut, M. Plantevit, C. Robardet, Local pattern detection in attributed graphs, in: Solving Large Scale Learning Tasks. Challenges and Algorithms, Springer, 2016, pp. 168–183.
[15] C. Loglisci, M. Ceci, D. Malerba, Relational mining for discovering changes in evolving networks, Neurocomputing 150 (2015) 265–288.
[16] S. Canu, A. Smola, Kernel methods and the exponential family, Neurocomputing 69 (7) (2006) 714–720.
[17] L. Song, K. Fukumizu, A. Gretton, Kernel embeddings of conditional distributions: a unified kernel framework for nonparametric inference in graphical models, IEEE Signal Process. Mag. 30 (4) (2013) 98–111.
[18] S. Yang, M. Wang, L. Jiao, Ridgelet kernel regression, Neurocomputing 70 (16) (2007) 3046–3055.
[19] R.I. Kondor, J. Lafferty, Diffusion kernels on graphs and other discrete input spaces, in: Proceedings of the Nineteenth International Conference on Machine Learning, ICML, 2, 2002, pp. 315–322.
[20] X. Zhu, Z. Ghahramani, Learning from labeled and unlabeled data with label propagation, Technical Report CMU-CALD-02-107.
[21] M.A. Lozano, F. Escolano, Graph matching and clustering using kernel attributes, Neurocomputing 113 (2013) 177–194.
[22] Z. Burda, J. Duda, J.-M. Luck, B. Waclaw, Localization of the maximal entropy random walk, Phys. Rev. Lett. 102 (16) (2009) 160602.
[23] R.-H. Li, J.X. Yu, J. Liu, Link prediction: the power of maximal entropy random walk, in: Proceedings of the Twentieth ACM International Conference on Information and Knowledge Management, ACM, 2011, pp. 1147–1156.
[24] J.K. Ochab, Z. Burda, Maximal entropy random walk in community detection, Eur. Phys. J. Spec. Top. 216 (1) (2013) 73–81.
[25] J.-C. Delvenne, A.-S. Libert, Centrality measures and thermodynamic formalism for complex networks, Phys. Rev. E 83 (4) (2011) 046117.
[26] R. Sinatra, J. Gómez-Gardenes, R. Lambiotte, V. Nicosia, V. Latora, Maximal-entropy random walks in complex networks with limited information, Phys. Rev. E 83 (3) (2011) 030103.
[27] K. Seeja, Feature selection based on closed frequent itemset mining: a case study on sage data classification, Neurocomputing 151 (2015) 1027–1032.
[28] J. Vreeken, N. Tatti, Interesting patterns, in: Frequent Pattern Mining, Springer, 2014, pp. 105–134.
[29] C. Faloutsos, V. Megalooikonomou, On data mining, compression, and Kolmogorov complexity, Data Min. Knowl. Discov. 15 (1) (2007) 3–20.
[30] H.T. Lam, F. Mörchen, D. Fradkin, T. Calders, Mining compressing sequential patterns, Stat. Anal. Data Min. ASA Data Sci. J. 7 (1) (2014) 34–52.
[31] Z. Wang, Y. Zhao, G. Wang, Y. Li, X. Wang, On extending extreme learning machine to non-redundant synergy pattern based graph classification, Neurocomputing 149 (2015) 330–339.
[32] M. Van Leeuwen, A. Siebes, StreamKrimp: detecting change in data streams, in: Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2008, pp. 672–687.
[33] J. Vreeken, M. Van Leeuwen, A. Siebes, Preserving privacy through data generation, in: Proceedings of the Seventh IEEE International Conference on Data Mining, ICDM, IEEE, 2007, pp. 685–690.
[34] J. Vreeken, A. Siebes, Filling in the blanks – Krimp minimisation for missing data, in: Proceedings of the Eighth IEEE International Conference on Data Mining, ICDM, IEEE, 2008, pp. 1067–1072.
[35] L. Akoglu, H. Tong, J. Vreeken, C. Faloutsos, Fast and reliable anomaly detection in categorical data, in: Proceedings of the Twenty-first ACM International Conference on Information and Knowledge Management, ACM, 2012, pp. 415–424.

[36] K. Smets, J. Vreeken, Slim: directly mining descriptive patterns, in: Proceedings of the SIAM International Conference on Data Mining, SIAM, 2012, pp. 236–247.
[37] O. Sampson, M.R. Berthold, Widened Krimp: better performance through diverse parallelism, in: Proceedings of the International Symposium on Intelligent Data Analysis, Springer, 2014, pp. 276–285.
[38] M. Dehmer, A. Mowshowitz, A history of graph entropy measures, Inf. Sci. 181 (1) (2011) 57–78.
[39] Y. Freund, L. Mason, The alternating decision tree learning algorithm, in: Proceedings of the Sixteenth International Conference on Machine Learning (ICML), 99, 1999, pp. 124–133.
[40] J. Platt, Sequential minimal optimization: A fast algorithm for training support vector machines, Technical Report MSR-TR-98-14.
[41] K.P. Murphy, Machine Learning: a Probabilistic Perspective, The MIT Press, 2012.
[42] E. Jones, T. Oliphant, P. Peterson, et al., SciPy: Open source scientific tools for Python (2001) [Online; accessed today] URL http://www.scipy.org/.
[43] A.H. Al-Mohy, N.J. Higham, A new scaling and squaring algorithm for the matrix exponential, SIAM J. Matrix Anal. Appl. 31 (3) (2009) 970–989.
[44] C. Moler, C. Van Loan, Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later, SIAM Rev. 45 (1) (2003) 3–49.
[45] A. Hagberg, D. Schult, P. Swart, D. Conway, L. Séguin-Charbonneau, C. Ellison, B. Edwards, J. Torrents, NetworkX: high productivity software for complex networks, Webová Stránka (2013). https://networkx.lanl.gov/wiki



Di Zhang is currently a Ph.D. student at the School of Computer Science, Communication University of China, Beijing, and also a researcher in Noah's Ark Lab, Huawei Corporation since 2011. He received the M.Sc. degree of Computer Science from the Beijing University of Aeronautics and Astronautics, China in 2006, and worked as research engineer in Institute of Software, Chinese Academy of Sciences from 2006 to 2010. His research interests include data mining, machine learning and distributed computing.