Decomposing a Sequence into Independent Subsequences Using Compression Algorithms

Hoang Thanh Lam
Technische Universiteit Eindhoven
2 Den Dolech
Eindhoven, the Netherlands
t.l.hoang@ie.ibm.com

Julia Kiseleva
Technische Universiteit Eindhoven
2 Den Dolech
Eindhoven, the Netherlands
j.kiseleva@tue.nl

Mykola Pechenizkiy
Technische Universiteit Eindhoven
2 Den Dolech
Eindhoven, the Netherlands
m.pechenizkiy@tue.nl

Toon Calders
Universite Libre de Bruxelles
CP 165/15 Avenue F.D. Roosevelt 50
B-1050 Bruxelles
toon.calders@ulb.ac.be

ABSTRACT

Given a sequence generated by a random mixture of independent processes, we study compression-based methods for decomposing the sequence into independent subsequences each corresponds to an independent process. We first show that the decomposition which results in the optimal compression length in expectation actually corresponds to an independent decomposition. This theoretical result encourages us to look for the decomposition that incurs the minimum description length to solve the independent decomposition problem. A hierarchical clustering algorithm is proposed to find that decomposition. We perform experiments with both synthetic and real-life datasets to show the effectiveness of our method in comparison with the state of the art method.

General Terms

Data mining

Keywords

Pattern mining, independent component decomposition, minimum description length principle, data compression

1. INTRODUCTION

Many processes produce extensive sequences of events, e.g. alarm messages from different components of industrial machines or telecommunication networks, web-access logs, clickstream data, geographical events record, etc. In many cases, a sequence consists of a random mixture of independent or loosely connected processes where each process produces a specific disjoint set of events that are independent from the other events.

It is useful to decompose the sequence into a number of independent subsequences. This data preprocessing step provides us with a lot of conveniences for further analysis with each independent process separately. In fact, independent sequence decomposition was used to improve the accuracy of predictive models by building local predictive model for each independent process separately instead of building it for the whole data [1, 2]. Besides, in descriptive data mining, people are usually interested in summarizing the data. They are eager to discover the dependency structure between events in the data and also want to know how strong the dependency is in each independent component [14]. If the dependency in a component is strong, it maybe associated with an explainable context that can help people understand the data better.

The sequence independent decomposition problem was first introduced by Mannila et al. in [3]. The authors proposed a method based upon a statistical hypothesis testing for the dependency between events. A drawback of the statistical hypothesis testing method is that the p-value derived from the test is a score subjective to the null hypothesis. The p-value does not convey any information about how strong the dependency between the events is. Moreover, the method introduces two parameters which require manual tunings for different applications. The method is also easily vulnerable to false detected connections between events (see the discussion in the next section for an example).

In this paper, we revisit the independent decomposition problem from the prospect of data compression. In order to illustrate the connection between data compression algorithms and the independent decomposition problem let us first consider a simple example. Assume that we have two sequences:

$$S_1 = abababababababababababababababab$$

$$S_2 = aaababbbabbaabbaababababaaabbbabab$$

The first sequence is very regular, after an occurrence of a there is an occurrence of b. In this case, it is clear that a and b are two dependent events. We can exploit this information to compress the sequence as follows: send the decoder the number of repetitions of ab, i.e. 16 times in S_1. Then we send the binary representations of a and b together. If the *Elias code* [4] is used to compress the natural number 16 and one bit is used to represent either a or b, the compressed size of S_1 is $9+1+1=11$ bits. On the other hand, if we do not exploit the dependency between a and b we need at least 32 bits to represent S_1. Therefore, by exploiting the knowledge

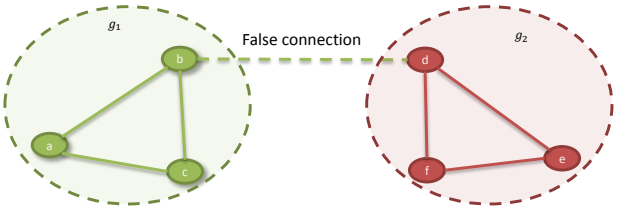

Figure 1: An example of two strongly connected and independent components. False connection between b and d is recognized just by chance or due to noise. The Dtest algorithm will merge two components together even only one false connection happens.

about the dependency between a and b we compress the sequence far better than the compression that considers a and b separately.

The second sequence seems like a random mixture between two independent events a and b. In this case, it does not matter how a compression algorithm does, the compression result will be very similar to the result of the compression algorithm that considers a and b separately.

In common-sense, two examples lead us to an intuition that if we can find the best way to compress the data then that compression algorithm may help us to reveal the dependency structure between events in a sequence simply because it will exploit that information for doing compression better. This intuition is inline with the general idea of the *Minimum Description Length* (MDL) principle [5] which always suggests that the best model is the one that describes the data in the shortest way. We get to the point to ask a fundamental question: What is the connection between the best model by the definition of the MDL principle [5] and the independent decomposition problem?

In this work, we study theoretical answers for the aforementioned question. In particular, we prove that the best model by the definition of the MDL principle actually corresponds to an independent decomposition of the sequence. This theoretical result motivates us to propose a data compression based algorithm to solve the independent decomposition problem by looking for a decomposition that can be used to compress the data most.

Beside being parameterless, the compression-based method provides us with a measure based on compression ratios showing how strong the connection between events of an independent component is. It can be considered as an interestingness measure to rank different independent components. We validate our method and compare it to the statistical hypothesis testing based approach [3] in an experiment with synthetic and real-life datasets.

2. RELATED WORKS

The sequence independent decomposition problem was first studied by Mannila et al. in [3]. The authors proposed a method based on statistical hypothesis testing for dependency between events. In this work, we call their method *Dtest* as for *Dependency Test*. The algorithm first performs dependency tests for every pair of events. Subsequently, it builds a dependency graph in which vertices correspond to events and edges connecting two dependent events.

An independent component of the output decomposition corresponds to a connected component of the graph. The Dtest approach has a drawback: it can merge two independent components together even when there is only one false connection (not a connection but erroneously detected as a connection) between two vertices across two components. For instance, Figure 1 shows two strongly connected components g_1 and g_2 of the dependency graph. If the dependency test between $b \in g_1$ and $d \in g_2$ produces wrong result, i.e. b and d pass the dependency test even they are independent, two independent components g_1 and g_2 will be merged into a single component.

In the experiment, we show that false connection is usually the case because of the following two reasons. First, the Dtest algorithm has two parameters. Setting of these parameters to avoid false connections is not always a trivial task. Second, the dependency between b and d can be incorrectly detected due to noises. Being different from the Dtest algorithm, our compression-based method is not easily vulnerable to false connections. Indeed, if two strongly connected components are independent to each other, the loose connection between b and d is not an important factor that can improve the compression ratio significantly when the two components are compressed together.

Indeed, independent component analysis for other types of data is a well studied problem in the literature [6]. For example, the ICA method was proposed to decompose a time series into independent components. However, the ICA method does not handle event sequence data.

Another closely related work concerns the item clustering problem studied under the context of itemset data [7]. The authors proposed a method to find clusters of strongly related items for data summarization. The work relies on the MDL principle which clusters items together such that it minimizes the description length of the data when the cluster structure is exploited for compressing the data. On one hand, our work proposes a different encoding to handle sequence data which is not handled by the encoding of [7]. On the other hand, we show a theoretical connection between the MDL principle and the independent component analysis. It gives a theoretical judgement for the model usually neutrally accepted as *the best* model by the definition of the MDL principle.

Finally, the idea of using data compression algorithms in data mining is not new. In fact, data compression algorithms were successfully used for many data mining tasks including data clustering [8] or data classification [9]. It was also used for mining non-redundant set of patterns in itemset data [10] and in sequence data [11]. Our work is the first one that proposes to use data compression for solving the independent sequence decomposition problem.

3. PROBLEM DEFINITION

Let $\sum = \{a_1, a_2, \cdots, a_N\}$ be the alphabet of events; denote $S_t = x_1 x_2 \cdots x_t$ as a sequence, where each $x_i \in \sum$ is generated by a random variable X_i ordered by its timestamp. The length of a sequence S is denoted as $|S|$.

We assume that S is generated by a stochastic process $\mathfrak{P}$. For any natural number n, we denote $P_t(X_t = x_t, X_{t+1} = x_{t+1}, \cdots, X_{t+n-1} = x_{t+n-1})$ as the joint probability of the sequence $X_{t+1}, X_{t+2}, \cdots, X_{t+n-1}$ governed by the stochastic process $\mathfrak{P}$, i.e. the probability of observing the subsequence $x_t x_{t+1} \cdots x_{t+n-1}$ at time point t.

A stochastic process is called *stationary* [4] if for any n the

joint probability $P_t(X_t = x_t, X_{t+1} = x_{t+1}, \cdots, X_{t+n-1} = x_{t+n-1})$ does not depend on t, which means that $P_t(X_t = x_t, X_{t+1} = x_{t+1}, \cdots, X_{t+n-1} = x_{t+n-1}) = P_1(X_1 = x_1, X_2 = x_2, \cdots, X_n = x_n)$ for any $t \geq 0$. In this work, we consider only stationary processes as in practice a lot of datasets are generated by a stationary process [4]. This assumption is made for convenience in our theoretical analysis although it is not a requirement for our algorithms to operate properly. Therefore, for a stationary process the joint probability $P_t(X_t, X_{t+2}, \cdots, X_{t+n-1})$ is simply denoted as $P(X_1, X_2, \cdots, X_n)$. For a given sequence S the probability of observing the sequence is simply denoted as $P(S)$.

Let $C = \{C_1, C_2, \cdots, C_k\}$ be a partition of the alphabet $\sum$ into k pairwise disjoint parts, where $C_i \bigcap C_j = \emptyset \ \forall i \neq j$ and $\bigcup_{i=1}^{k} C_i = \sum$. Given a sequence S, the partition C decomposes S into k disjoint subsequences denoted as $S(C_i)$ for $i = 1, 2, \cdots, k$. Let $P_i(S(C_i) = s)$ denote the marginal distribution defined on a set of subsequence with fixed size $|s| < |S|$.

EXAMPLE 1. *Let the alphabet $\sum = \{a, b, c, d, e, f, g\}$ be partitioned into three disjoint parts: $C = \{C_1, C_2, C_3\}$ where $C_1 = \{a, b, c\}$, $C_2 = \{d, e\}$ and $C_3 = \{f, g\}$. The partition C decomposes the sequence $S = abdffeadcdeabgg$ into three subsequences $S(C_1) = abacab$, $S(C_2) = dedde$ and $S(C_3) = ffgg$.*

Denote α_i as the probability of observing an event belonging to the cluster C_i. Assume that $\mathfrak{P}_i$ is the stochastic process that generates $S(C_i)$.

DEFINITION 1 (INDEPENDENT DECOMPOSITION). *We say that $C = \{C_1, C_2, \cdots, C_k\}$ is an independent decomposition of the alphabet if S is a random mixture of independent subsequences $S(C_i)$:*

$$P\left(S\left(\bigcup_{i=1}^{k} C_i\right)\right) = \prod_{i=1}^{k} \alpha_i^{|S(C_i)|} P_i\left(S(C_i)\right)$$

There are many independent decompositions, we are interested in the decomposition with maximum k; denote that decomposition as C^*. The problem of independent sequence decomposing can be formulated as follows:

DEFINITION 2 (SEQUENCE DECOMPOSITION). *Given a sequence S and an alphabet $\sum$, find the maximum independent decomposition C^* of S.*

THEOREM 1 (UNSOLVABLE). *Observing a sequence generated by a stochastic (stationary) process with bounded size M there is no deterministic algorithm that solves the sequence independent decomposition problem exactly.*

PROOF. Assume that there is a deterministic algorithm A that can return the maximum independent decomposition exactly when up to $2 * M$ events of a sequence are observed. Consider the following alphabet $\sum = \{a, b\}$ and two different stationary processes:

- The events a and b are drawn independently at random with probability 0.5
- The events a and b are drawn from a simple Markov chain with two states a and b and $P(a \mapsto b) = P(b \mapsto a) = 1.0$

The sequence $S = (ab)^M$ with length $2M$ can be generated by both stationary processes with non-zero probability. Therefore, by observing S, the algorithm A cannot decide the maximum independent decomposition C^* because for the latter process $C^* = \{\{a, b\}\}$ while for the former process $C^* = \{\{a\}, \{b\}\}$. This point leads to contradiction. □

4. SEQUENCE COMPRESSION

Given an observed sequence with bounded size, Theorem 1 shows that the problem in Definition 2 is unsolvable. However, in this section we show that it can be solved asymptotically by using data compression algorithms. We first define encodings that we use to compress a sequence S given a decomposition $C = \{C_1, C_2, \cdots, C_k\}$.

Given an event a_i denote $I(a_i) = j$ as the identifier of the cluster (partition) C_j which contains a_i. Let $S = x_1 x_2 \cdots x_n$ be a sequence, denote $I(S)$ as the cluster identifier sequence, i.e. $I(S) = I(x_1)I(x_2)\cdots I(x_n)$.

EXAMPLE 2. *In Example 1, given the decomposition $C_1 = \{a, b, c\}$, $C_2 = \{d, e\}$ and $C_3 = \{f, g\}$ the cluster identifier sequence of S is $I(abdffeadcdeabgg) = 112332121221133$.*

If the distribution of the cluster identifiers is given as $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_k)$, where $\sum_{j=1}^{k} \alpha_j = 1$, the *Huffman code* [4] can be used to encode each cluster identifier j in the sequence $I(S)$ with a codeword with length proportional to $-\log \alpha_j$. In expectation, if the identifiers are independent to each other that encoding results in the minimum compression length for the cluster identifier sequence [4]. Denote $E^*(I(S))$ as the encoded form of $I(S)$ in that *ideal encoding.*

In practice, we don't know the distribution $(\alpha_1, \alpha_2, \cdots, \alpha_k)$. However, the distribution can be estimated from data. An encoding is called *asymptotically optimal* if:

$$\lim_{S \mapsto \infty} \frac{|E^+(I(S))|}{|S|} = H(\alpha)$$

Where $H(\alpha)$ denotes the entropy of the distribution α. An example of E^+ is the one that uses the empirical value $\frac{|S(C_i)|}{|S|}$ as an estimate of α_i.

Let Z be a data compression algorithm that gets the input as a sequence and returns the compressed form of that sequence. Z is an *ideal compression algorithm* denoted as Z^* if $|Z^*(S)| = -\log P(S)$. In expectation, Z^* results in the minimum compression length for the data [4]. In practice, we don't know the distribution $P(S)$ however we can use an asymptotic approximation of the ideal compression algorithm, e.g. the *Lempel-Ziv algorithms* [4]. An algorithm is asymptotically optimal denoted as $Z^+(S)$ if $\lim_{S \mapsto \infty} \frac{|Z^+(S)|}{|S|} = H(\mathfrak{P})$, where $\mathfrak{P}$ is the stationary process that generates S.

Given a decomposition $C = \{C_1, C_2, \cdots, C_k\}$ and a compression algorithm Z, the sequence S can be compressed in two parts, the first part corresponds to the compressed form of the identifier sequence $I(S)$. The second part contains k compressed subsequences $Z(S(C_1)), Z(S(C_2)), \cdots, Z(S(C_k))$. In summary, the compressed sequence consists of the size of the sequence in an encoded form denoted as $E(|S|)$, the compressed form of the cluster identifier sequence and k compressed subsequences. In this work, we use the term *ideal encoding* to refer to the encoding that uses E^* and Z^* and *asymptotic encoding* to refer to the encoding that uses E^+ and Z^+.

EXAMPLE 3. *In Example 1, given the decomposition* $C_1 = \{a, b, c\}$, $C_2 = \{d, e\}$ *and* $C_3 = \{f, g\}$, *the sequence* S *in Example 1 can be encoded as follows:* $E(15)\ E(I(S))\ Z(S(C_1))\ Z(S(C_2))\ Z(S(C_3))$

5. THE MDL PRINCIPLE AND INDEPENDENT DECOMPOSITIONS:

The description length of the sequence S using the decomposition C can be calculated as: $L^C(S) = |E(|S|)| + |E(I(S))| + \sum_{i=1}^{k} |Z(S(C_i))|$. In this encoding, the term $|E(|S|)|$ is invariant when S is given. The decomposition C can be considered as a model and the cost to describe that model is equal to the term $|E(I(S))|$, meanwhile the latter term $\sum_{i=1}^{k} |Z(S(C_i))|$ corresponds to cost of describing the data given the model C. Therefore, according to the minimum description length principle we may try to find a decomposition resulting in the minimum description length in expectation, which is believed to be the best model for describing the data.

This section introduces two key theoretical results: subsection 5.1 shows an ideal analysis that given the data with bounded size, under an ideal encoding the best model describing the data corresponds to an independent decomposition and vice versa. Subsection 5.2 discusses an asymptotic result showing that under an asymptotic encoding, any independent decomposition corresponds to the best model by the definition of the MDL principle.

5.1 Analysis under the ideal encoding

We recall some definitions in information theory. Given a discrete probability distribution $P = \{\alpha_1, \alpha_2, \cdots, \alpha_k\}$ where $\sum_{i=1}^{k} \alpha_i = 1$, the *entropy* of the distribution P denoted as $H(P)$ is calculated as $-\sum_{i=1}^{k} \alpha_i \log \alpha_i$.

Given a stochastic process $\mathfrak{P}$ which generates the sequence S, denote $H(\mathfrak{P})$ as the *entropy rate* or entropy for short of the stochastic process $\mathfrak{P}$. Recall that $H(\mathfrak{P})$ is defined as $\lim_{n\mapsto\infty} \frac{1}{n} H(X_1, X_2, \cdots, X_n)$, where $H(X_1, X_2, \cdots, X_n)$ stands for the joint entropy of the random variables $X_1, X_2, \cdots, X_n$. It has been shown that when $\mathfrak{P}$ is a stationary process $\lim_{n\mapsto\infty} \frac{1}{n} H(X_1, X_2, \cdots, X_n)$ exists [4].

THEOREM 2 (MDL VS. INDEPENDENT DECOMPOSITIO). *Under an ideal encoding, given data with bounded size, the best decomposition which results in the minimum data description length in expectation is an independent decomposition and vice versa.*

PROOF. Given a decomposition $C = \{C_1, C_2, \cdots, C_k\}$, for a given n assume that S is a sequence with length n. Under an ideal encoding, the description length of the cluster identifier sequence of S is $|E^*(I(S))| = -\sum_{i=1}^{k} |S(C_i)| \log \alpha_i$. In the ideal encoding, since the length of the compressed subsequence $Z^*(S(C_i))$ is $|Z^*(S(C_i))| = -\log P_i(S(C_i))$ the total description length is:

$$
\begin{aligned}
L^C(S) &= |E(n)| - \sum_{i=1}^{k} |S(C_i)| \log \alpha_i && (1)\\
&\quad - \sum_{i=1}^{k} \log P_i(S(C_i)) && (2)
\end{aligned}
$$

$$
\begin{aligned}
E(L^C(S)) &= \sum_{|S|=n} P(S) * L^C(S) && (3)\\
&= |E(n)| - \sum_{|S|=n} P(S) && (4)\\
&\quad \log \prod_{i=1}^{k} \alpha_i^{|S(C_i)|} P_i(S(C_i)) && (5)\\
&= |E(n)| + H_P(X_1, X_2, \cdots, X_n) + && (6)\\
&\quad D(P|Q) && (7)
\end{aligned}
$$

Where Q is the random mixture of the distributions P_i defined on the space of all sequence $S : |S| = n$, i.e. $Q(S) = \prod_{i=1}^{k} \alpha_i^{|S(C_i)|} P_i(S(C_i))$ and $D(P|Q)$ is the relative entropy or the *Kullback-Leibler* distance between P and Q. Since $D(P|Q) \geq 0$ [4] we can imply that $E(L^C(S)) \geq |E(n)| + H_P(X_1, X_2, \cdots, X_n)$. The equality happens if and only if $D(P|Q) = 0$, i.e. $P \equiv Q$ which proves the theorem. □

5.2 Analysis under the asymptotic encoding

In the ideal analysis, it requires an ideal encoding which is not a practical assumption. However, we can still prove a similar result under an asymptotic encoding. First, we prove a basic supporting lemma. The lemma is a generalized result of the *Cesàro mean* [12].

LEMMA 1. *Given a sequence* (a_n), *a sequence* (c_n) *is defined as :* $c_n = \sum_{i=1}^{n} b_i(n) a_i$ *where* $\sum_{i=1}^{n} b_i(n) = 1$ *and* $b_i(n) > 0$ $\forall n > 0$. *If* $\lim_{n\mapsto\infty} a_n = A$ *and* $\lim_{n\mapsto\infty} b_i(n) = 0\ \forall i > 0$ *then we also have* $\lim_{n\mapsto\infty} c_n = A$.

PROOF. Since $\lim_{n\mapsto\infty} a_n = A$ given any number $\epsilon > 0$ there exists N such that $|a_n - A| < \frac{\epsilon}{2}\ \forall n > N$. Moreover, because $\lim_{n\mapsto\infty} a_n = A$ there exists an upper bound D on $|a_i - A|$.

Given N, since $\lim_{n\mapsto\infty} b_i(n) = 0$ we can choose M_i $(i = 1, 2, \cdots, N)$ such that $b_i(n) < \frac{\epsilon}{2ND}\ \forall n > M_i$. Let denote M as the maximum value of the set $\{N, M_1, M_2, \cdots, M_N\}$. For any $n > M$, we have:

$$
\begin{aligned}
|c_n - A| &= |\sum_{i=1}^{n} b_i(n) a_i - A| && (8)\\
&\leq |\sum_{i=1}^{N-1} b_i(n)(a_i - A)| + && (9)\\
&\quad |\sum_{i=N}^{n} b_i(n)(a_i - A)| && (10)\\
&\leq (N-1) D \frac{\epsilon}{2ND} + \frac{\epsilon}{2} && (11)\\
&\leq \epsilon && (12)
\end{aligned}
$$

The last inequality proves the lemma. □

THEOREM 3 (INDEPENDENT DECOMPOSITION ENTROPY). *Assume that* $C = \{C_1, C_2, \cdots, C_k\}$ *is an independent decomposition and* $\mathfrak{P}_\mathfrak{i}$ *is the stochastic process that generates* $S(C_i)$. *Denote* α_i *as the probability that we observe an event belonging to the cluster* C_i, *we have:*

$$
H(\mathfrak{P}) = \sum_{i=1}^{k} \alpha_i H(\mathfrak{P}_\mathfrak{i}) + H(\alpha_1, \alpha_2, \cdots, \alpha_k) \qquad (13)
$$

PROOF. We first prove a special case with $k = 2$ from which the general case for any k can be directly implied. Denote $H(X_1, X_2, \cdots, X_n)$ as H for short. In fact, by the definition of the joint entropy we can perform simple calculations as follows:

$$\begin{aligned} H &= -\sum_{|S|=n} P(S)\log P(S) && (14)\\ &= -\sum_{|S|=n} \alpha_1^{|S(C_1)|}P_1(S(C_1))\alpha_2^{|S(C_2)|}P_2(S(C_2)) && (15)\\ &\quad \log\left(\alpha_1^{|S(C_1)|}P_1(S(C_1))\alpha_2^{|S(C_2)|}P_2(S(C_2))\right) && (16)\\ &= -C_n^{|S_1|}\sum_{|S_1|\le n}\sum_{|S_2|=n-|S_1|}\alpha_1^{|S_1|}P_1(S_1) && (17)\\ &\quad \alpha_2^{|S_2|}P_2(S_2)\log\left(\alpha_1^{|S_1|}P_1(S_1)\alpha_2^{|S_2|}P_2(S_2)\right) && (18)\end{aligned}$$

We denote each term of Equation 18 as follows:

$$\begin{aligned} X &= -C_n^{|S_1|}\sum_{|S_1|\le n}\sum_{|S_2|=n-|S_1|}\alpha_1^{|S_1|}P_1(S_1)\alpha_2^{|S_2|} && (19)\\ &\quad P_2(S_2)\log\alpha_1^{|S_1|} && (20)\end{aligned}$$

$$\begin{aligned} Y &= -C_n^{|S_1|}\sum_{|S_1|\le n}\sum_{|S_2|=n-|S_1|}\alpha_1^{|S_1|}P_1(S_1)\alpha_2^{|S_2|} && (21)\\ &\quad P_2(S_2)\log\alpha_2^{|S_2|} && (22)\end{aligned}$$

$$\begin{aligned} Z &= -C_n^{|S_1|}\sum_{|S_1|\le n}\sum_{|S_2|=n-|S_1|}\alpha_1^{|S_1|}P_1(S_1)\alpha_2^{|S_2|} && (23)\\ &\quad P_2(S_2)\log P_1(S_1) && (24)\end{aligned}$$

$$\begin{aligned} T &= -C_n^{|S_1|}\sum_{|S_1|\le n}\sum_{|S_2|=n-|S_1|}\alpha_1^{|S_1|}P_1(S_1)\alpha_2^{|S_2|} && (25)\\ &\quad P_2(S_2)\log P_2(S_2) && (26)\end{aligned}$$

We calculate each term of Equation 18 as follows:

$$\begin{aligned} X &= -\sum_{i=0}^{n}C_n^i\sum_{|S_1|=i}\sum_{|S_2|=n-i}\alpha_1^i\alpha_2^{n-i} && (27)\\ &\quad P_1(S_1)P_2(S_2)\log\alpha_1^i && (28)\\ &= -\sum_{i=0}^{n}C_n^i\alpha_1^i\alpha_2^{n-i}\log\alpha_1^i && (29)\\ &\quad \sum_{|S_1|=i}\sum_{|S_2|=n-i}P_1(S_1)P_2(S_2) && (30)\\ &= -\sum_{i=0}^{n}C_n^i\alpha_1^i\alpha_2^{n-i}\log\alpha_1^i && (31)\\ &= -\log\alpha_1\sum_{i=0}^{n}iC_n^i\alpha_1^i\alpha_2^{n-i} && (32)\\ &= -n\alpha_1\log\alpha_1 && (33)\end{aligned}$$

With similar calculation we have $Y = -n\alpha_2\log\alpha_2$. We continue with the calculation of Z:

$$\begin{aligned} Z &= -\sum_{i=0}^{n}C_n^i\sum_{|S_1|=i}\sum_{|S_2|=n-i}\alpha_1^i\alpha_2^{n-i} && (34)\\ &\quad P_1(S_1)P_2(S_2)\log P_1(S_1) && (35)\\ &= -\sum_{i=0}^{n}C_n^i\sum_{|S_1|=i}\alpha_1^i\alpha_2^{n-i}P_1(S_1)\log P_1(S_1) && (36)\\ &\quad \sum_{|S_2|=n-i}P_2(S_2) && (37)\\ &= -\sum_{i=0}^{n}C_n^i\sum_{|S_1|=i}\alpha_1^i\alpha_2^{n-i}P_1(S_1)\log P_1(S_1) && (38)\\ &= -\sum_{i=0}^{n}C_n^i\alpha_1^i\alpha_2^{n-i}\sum_{|S_1|=i}P_1(S_1)\log P_1(S_1) && (39)\\ &= \sum_{i=1}^{n}C_n^i\alpha_1^i\alpha_2^{n-i}H_{P_1}(X_1, X_2, \cdots, X_i) && (40)\end{aligned}$$

With similar calculation we also have:

$$T = \sum_{i=1}^{n}C_n^i\alpha_1^{n-i}\alpha_2^i H_{P_2}(X_1, X_2, \cdots, X_i)$$

Therefore we further imply that:

$$\begin{aligned} H &= X + Y + Z + T && (41)\\ &= nH(\alpha_1, \alpha_2) + && (42)\\ &\quad \sum_{i=1}^{n}C_n^i\alpha_1^i\alpha_2^{n-i} * H_{P_1}(X_1, X_2, \cdots, X_i) + && (43)\\ &\quad \sum_{i=1}^{n}C_n^i\alpha_1^{n-i}\alpha_2^i H_{P_2}(X_1, X_2, \cdots, X_i) && (44)\end{aligned}$$

$$\begin{aligned} \frac{H}{n} &= H(\alpha_1, \alpha_2) + && (45)\\ &\quad \alpha_1\sum_{i=1}^{n}C_{n-1}^{i-1}\alpha_1^{i-1}\alpha_2^{n-i}\frac{1}{i}H_{P_1}(X_1, \cdots, X_i) + && (46)\\ &\quad \alpha_2\sum_{i=1}^{n}C_{n-1}^{i-1}\alpha_1^{n-i}\alpha_2^{i-1}\frac{1}{i}H_{P_2}(X_1, X_2, \cdots, X_i) && (47)\end{aligned}$$

Besides, we have:

$$\lim_{n\mapsto\infty}\frac{1}{n}H(X_1, \cdots, X_n) = H(\mathfrak{P})$$
$$\lim_{n\mapsto\infty}\frac{1}{n}H_{P_1}(X_1, \cdots, X_n) = H(\mathfrak{P}_1)$$
$$\lim_{n\mapsto\infty}\frac{1}{n}H_{P_2}(X_1, \cdots, X_n) = H(\mathfrak{P}_2)$$

Therefore, according to Lemma 1 from the last equation we can imply that $H(\mathfrak{P}) = \alpha_1 H(\mathfrak{P}_1) + \alpha_2 H(\mathfrak{P}_2) + H(\alpha_1, \alpha_2)$.

The last result can be easily generalized for an independent decomposition with any k clusters by induction. Indeed, we assume that the theorem is correct with $k = l$ we prove that the result holds for $k = l + 1$. Denote α as $\sum_{i=1}^{l}\alpha_i$. Given two independent stochastic processes $\mathfrak{P}$ and $\mathfrak{Q}$ denote the random mixture of them as $\mathfrak{P}\bigoplus\mathfrak{Q}$. Consider the process defined as the random mixture of $\mathfrak{P}_1, \mathfrak{P}_2 \cdots \mathfrak{P}_l$ denoted as $\mathfrak{P}_1\bigoplus\mathfrak{P}_2\bigoplus\cdots\bigoplus\mathfrak{P}_l$. Since $\mathfrak{P}_{l+1}$ and the se-

quence $\mathfrak{P}_1 \bigoplus \mathfrak{P}_2 \bigoplus \cdots \bigoplus \mathfrak{P}_l$ are independent we have:

$$H(\mathfrak{P}) = H(\mathfrak{P}_1 \bigoplus \mathfrak{P}_2 \bigoplus \cdots \bigoplus \mathfrak{P}_{l+1}) \quad (48)$$

$$= \alpha H(\mathfrak{P}_1 \bigoplus \mathfrak{P}_2 \bigoplus \cdots \bigoplus \mathfrak{P}_l) + \quad (49)$$

$$\alpha_{l+1} H(\mathfrak{P}_{l+1}) + H(\alpha, \alpha_{l+1}) \quad (50)$$

Moreover, by the induction assumption:

$$H(\mathfrak{P}_1 \bigoplus \mathfrak{P}_2 \bigoplus \cdots \bigoplus \mathfrak{P}_l) = \sum_{i=1}^{k} \frac{\alpha_i}{\alpha} H(\mathfrak{P}_i) + \quad (51)$$

$$H(\frac{\alpha_1}{\alpha}, \frac{\alpha_2}{\alpha}, \cdots, \frac{\alpha_l}{\alpha}) \quad (52)$$

Replacing this value to Equation 50, we can obtain Equation 13 from which the theorem is proved. □

Theorem 3 shows that the entropy of the stochastic process $\mathfrak{P}$ can be represented as the sum of two meaningful terms. The first term $H(\alpha_1, \alpha_2, \cdots, \alpha_k)$ actually corresponds the average cost per element of the cluster identifier. Meanwhile the second term $\sum_{i=1}^{k} \alpha_i H(\mathfrak{P}_i)$ corresponds to the average cost per element to encode the subsequences $S(C_i)$. By that important observation we can show the following asymptotic result:

THEOREM 4 (ASYMPTOTIC RESULT). *Under an asymptotical encoding, the data description length in an independent decomposition is asymptotically optimal with probability equal to 1.*

PROOF. Let $C = \{C_1, C_2, \cdots, C_k\}$ be an independent decomposition, for any n assume that S is a sequence with length n. Under an asymptotic encoding, the description length of the data is:

$$L^C(S) = |E(n)| + |E^+(I(S))| + \sum_{i=1}^{k} Z^+(S(C_i)) \quad (53)$$

$$\frac{L^C(S)}{|S|} = \frac{|E(n)|}{|S|} + \frac{|E^+(I(S))|}{|S|} + \frac{\sum_{i=1}^{k} Z^+(S(C_i))}{|S|} \quad (54)$$

$$\frac{L^C(S)}{|S|} = \frac{|E(n)|}{|S|} + \frac{|E^+(I(S))|}{|S|} + \sum_{i=1}^{k} \frac{|S(C_i)|}{|S|} \frac{Z^+(S(C_i))}{|S(C_i)|} \quad (55)$$

$$Pr\left(\lim_{|S| \mapsto \infty} \frac{L^C(S)}{|S|} = H(\alpha_1, \alpha_2, \cdots, \alpha_k) + \sum_{i=1}^{k} \alpha_i H(\mathfrak{P}_i) \right) = 1 \quad (56)$$

$$Pr\left(\lim_{|S| \mapsto \infty} \frac{L^C(S)}{|S|} = H(\mathfrak{P}) \right) = 1 \quad (57)$$

The last equation is a direct result of Theorem 3. Since $H(\mathfrak{P})$ is the lower-bound on the expectation of the average compression size per element of any data compression algorithm the encoding using the independent decomposition is asymptotically optimal. □

The ideal analysis shows the one-to-one correspondence between the optimal encoding and an independent decomposition. The asymptotic result only shows that an independent decomposition asymptotically corresponds to an optimal encoding. The theorem does not prove the reverse correspondence; however, in experiments we empirically show that the correspondence is one-to-one.

Algorithm 1 Dzip(S)

```
1: Input: a sequence S, an alphabet ∑ = {a_1 a_2 ··· a_N}
2: Output: a decomposition C
3: C ← {C_1 = {a_1}, C_2 = {a_2}, ··· , C_n = {a_n}}
4: while true do
5:   max ← 0
6:   C* ← C
7:   for i = 1 to |C| do
8:     for j = i + 1 to |C| do
9:       C+ ← C with merged C_i and C_j
10:      if L^C(S) − L^{C+}(S) > max then
11:        max ← L^C(S) − L^{C+}(S)
12:        C* ← C+
13:      end if
14:    end for
15:  end for
16:  if |C*| = 1 or max = 0 then
17:    Return C*
18:  end if
19: end while
```

6. ALGORITHMS

The theoretical analysis in Section 5 encourages us to design an algorithm that looks for the best decomposition to find an independent decomposition. When an independent decomposition is found, the algorithm can be recursively repeated on each independent component to find the maximum independent decomposition. Given data S with alphabet $\sum$ this section discusses a hierarchical clustering algorithm called Dzip to find the desired decomposition.

Algorithm 1 shows the main steps of the Dzip algorithm. It starts with N clusters each contains only one character of the alphabet. Subsequently, it evaluates the compression benefit of merging any pair of clusters. The best pair of clusters which results in the smallest compression size is chosen to be merged together. These steps are repeated until there is no compression benefit of merging any two clusters or all the characters are already merged into a single cluster.

Dzip can be recursively applied on each cluster to get the maximum decomposition. However, in our experiment we observe that in most of the cases the cluster found by Dzip cannot be decomposed further because of the bottom-up process which already checks for the benefits of splitting the cluster.

Dzip uses the *Lempel-Ziv-Welch* (LZW) implementation [4] with complexity linear in the size of the data. It utilizes an inverted list data structure to store the list of positions of each character in the sequence. Moreover, it also caches compression size of merged clusters. In doing so, in the worst case the computational complexity of Dzip can be bounded as $O(|S|N^2)$. This number is the same as the amortized complexity of the Dtest algorithm [3].

7. EXPERIMENTS

We consider the dependency test method Dtest [3] as a baseline approach. All the experiments were carried out on a 16 processor cores, 2 Ghz, 12 GB memory, 194 GB local disk, Fedora 14 / 64-bit machine. The source codes of Dtest and Dzip in Java and the datasets are available for download

in our project website[1].

Dtest has two parameters: the significance value α and the gap number G. We choose $\alpha = 0.01$ and $G = 300$ as recommended in the paper [3]. We also tried to vary α and G from small to large and observed quite different results. The algorithm is slow when G increases, while smaller value of α results in low false positive yet high false negative rate and vice versa. However, the results with different parameters do not change the comparisons in our experiments.

7.1 Synthetic data

There are three datasets in this category for which the ground truths are known:

- *Parallel:* is a synthetic dataset which mimics a typical situation in practice where the data stream is generated by five independent parallel processes. Each process P_i $(i = 1, 2, \cdots, 5)$ generates one event from the set of events $\{A_i, B_i, C_i, D_i, E_i\}$ in that order. In each step, the generator chooses one of five processes uniformly at random and generates an event by using that process until the stream length is 1M.

- *Noise:* is generated in the same way as the parallel dataset but with additional noises. A noise source generates independent events from a noise alphabet with size 1000. Noise events are randomly mixed with parallel dataset. The amount of noises is 20% of the parallel data. This dataset is considered to see how the methods will be sensitive to noise.

- *HMM:* is generated by a random mixture of two different hidden Markov models. Each hidden Markov model has 10 hidden states and 5 observed states. The transition matrix and the emission matrix are generated randomly according to the standard normal distribution with mean in the diagonal of the matrix. Each Markov model generates 5000 events and the mixture of them contains 10000 events. This dataset is considered to see the performance of the methods in a small dataset.

Since the ground-truths are known we use the *Rand index* [13] to compare two partitions of the alphabet set. The rand index calculates the number of pairs of events that are either in the same cluster in both partitions or in different clusters in both partitions. This number is normalized to have value the interval $[0, 1]$ by dividing by the total number of different pairs. The rand index measures the agreement between two different partitions, where a value of 1 means a perfect match, while 0 means two partitions completely disagree to each other.

Figure 2 shows rand index (y-axis) of two algorithms when the data size (x-axis) is varied. It is clear from the figure that the Dzip algorithm is possible to return a perfect decomposition in all datasets. When the data size is smaller the performance is slightly changed but the rand index is still high.

The performance of the Dtest algorithm is good in the Parallel dataset although the result is not stable when the datasize varied. However, Dtest does not work well in the Noise and the HMM dataset especially when a lot of noises are added. In both datasets, Dtest seems to cluster every events together; this experiment confirms our discussion in section 2 that Dtest is vulnerable to noise.

[1] `www.win.tue.nl/~lamthuy/dzip.htm`

7.2 Real-life data

There are three datasets in this category:

- *Machine:* is a message log containing about 2.7 million of messages (more than 1700 distinct message types) produced by different components of a photolithography machine.

- *MasterPortal:* is a historical log of user behaviors in the *MasterPortal*[2] website. It contains about 1.7M of events totally of 16 different types of behaviors such as *Program view, University view, Scholarship view, Basic search, Click on ads banner* and so on.

- *Msnbc:* is the clickstream log by the users of the MSNBC website[3]. The log contains 4.6M events of 16 different types each corresponds to a category of the website such as *frontpage, sports, news, technology, weather* and so on.

For the Machine dataset, the Dzip algorithm produced 11 clusters with three major clusters contains a lots of events. Meanwhile, the Dtest algorithm produced 4 clusters with one very big cluster and 3 outlier clusters each contains only one event. The result shows that Dtest seems to cluster every events together. Since the ground truths are unknown we compare the compression ratios when using the decomposition by each algorithm to compress the data. Our observation shows that the data is compressed better with the decomposition produced by the Dzip algorithm because the compression ratio is 2.37 on the Dzip algorithm versus 2.34 on the Dtest algorithm.

Both Dzip and Dtest produced one cluster of events for the Msnbc and the MasterPortal datasets. Therefore, the compression ratios of both algorithms are the same in each dataset. Although we don't know the ground-truths for these datasets the result seems to be reasonable because in the case of clickstream data, users traverse on the web graphs and the relations between the events are inherently induced from the connections in the web graph structure. The Msnbc is compressed better than the MasterPortal (the compression ratios are 1.5 and 1.2 respectively). These numbers also tell us that the dependency between events in the Msnbc dataset seems to be more regular.

7.3 Running time

In section 6, we have shown that the amortized complexity of the Dtest algorithm and the worst case complexity of the Dzip algorithm are the same. The result promises that the Dzip algorithm will be faster than the Dtest algorithm. Indeed, this fact holds for the set of datasets we use in this paper. In Figure 3 we compare Dzip and Dtest in terms of running time. In most datasets, Dzip is about an order of magnitude faster than the Dtest algorithm.

8. CONCLUSIONS AND FUTURE WORKS

In this paper, we proposed a compression-based method called Dzip for the sequence independent decomposition problem. Beside being justified by a theoretical analysis, in experiments with both synthetic and real-life datasets, Dzip

[2] `www.mastersportal.eu`

[3] `www.msnbc.com`

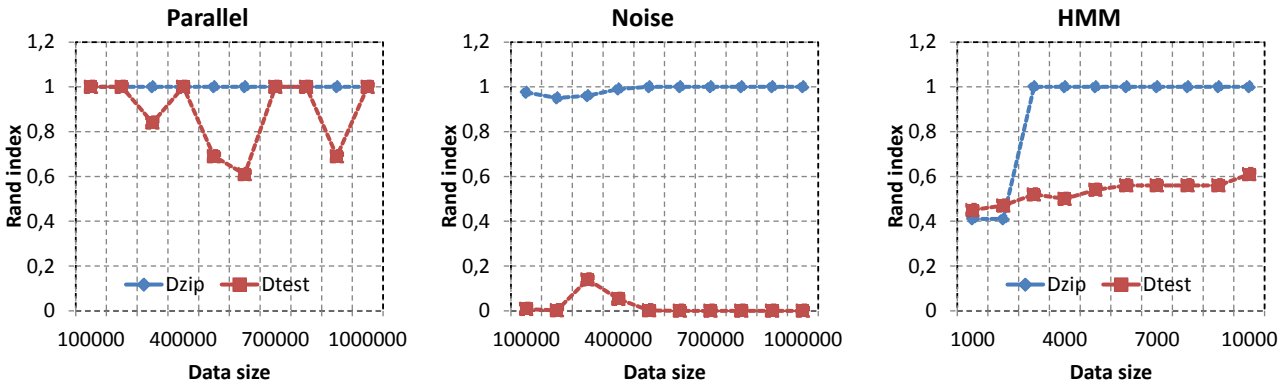

Figure 2: Rand index measures the similarity (higher is better) between the decompositions by the algorithms and the ground-truths. Rand index is equal to 1 if it is a perfect decomposition.

	MasterPortal	Msnbc	Machine	Parallel	Noise	HMM
Dzip	28	38	131387	8	192	1
Dtest	159	500	201385	138	25130	1

Figure 3: Running time in seconds of two algorithms. Dzip is about an order of magnitude faster than Dtest.

was shown to be more effective than the state of the art method based on statistical hypothesis testing. There are various directions to extend the paper for the future works. At the moment, we assume that each independent process must produce a disjoint subset of events. In practice, the case that independent processes produce overlapping subset of events is not rare. Extending the work to this more general case can be considered as an interesting future work.

9. ACKNOWLEDGEMENTS

The work is part of the project *Mining Complex Patterns in Stream* (COMPASS) supported by the *Netherlands Organisation for Scientific Research* (NWO).

10. REFERENCES

[1] Kira Radinsky, Eric Horvitz: Mining the web to predict future events. WSDM 2013: 255-264

[2] Julia Kiseleva, Hoang Thanh Lam, Toon Calders and Mykola Pechenizkiy: Discovery temporal hidden contexts in web sessions for user trail prediction TempWeb workshop at WWW 2013.

[3] Heikki Mannila, Dmitry Rusakov: Decomposition of event sequences into independent components. SDM 2001

[4] Thomas Cover, Joy Thomas: Elements of information theory. Wiley and Son, second edition 2006.

[5] Peter Grünwald: The minimum description length principle. MIT press, 2007.

[6] Hyvärinen A, Oja E. Independent component analysis: algorithms and applications. Journal of Neural Network 2000.

[7] Michael Mampaey, Jilles Vreeken: Summarizing categorical data by clustering attributes. Data Min. Knowl. Discov. 2013

[8] Rudi Cilibrasi, Paul M. B. Vitányi: Clustering by compression. IEEE Transactions on Information Theory 2005

[9] Eamonn J. Keogh, Stefano Lonardi, Chotirat Ann Ratanamahatana, Li Wei, Sang-Hee Lee, John Handley: Compression-based data mining of sequential data. Data Min. Knowl. Discov. 2007

[10] Jilles Vreeken, Matthijs van Leeuwen, Arno Siebes: Krimp: mining itemsets that compress. Data Min. Knowl. Discov. 2011

[11] Hoang Thanh Lam, Fabian Moerchen, Dmitriy Fradkin, Toon Calders: Mining Compressing Sequential Patterns. SDM 2012: 319-330

[12] Hardy, G. H. (1992). Divergent Series. Providence: American Mathematical Society. ISBN 978-0-8218-2649-2.

[13] W. M. Rand (1971). Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association 66 (336).

[14] van der Aalst, W. M. P., Weijters, T. and Maruster, L. (2004). Workflow Mining: Discovering Process Models from Event Logs.. IEEE Trans. Knowl. Data Eng., 16, 1128-1142.