Generating Realistic Synthetic Population Datasets

HAO WU, Virginia Tech
YUE NING, Virginia Tech
PRITHWISH CHAKRABORTY, Virginia Tech
JILLES VREEKEN, Max Planck Institute for Informatics and Saarland University
NIKOLAJ TATTI, Aalto University
NAREN RAMAKRISHNAN, Virginia Tech

Modern studies of societal phenomena rely on the availability of large datasets capturing attributes and activities of synthetic, city-level, populations. For instance, in epidemiology, synthetic population datasets are necessary to study disease propagation and intervention measures before implementation. In social science, synthetic population datasets are needed to understand how policy decisions might affect preferences and behaviors of individuals. In public health, synthetic population datasets are necessary to capture diagnostic and procedural characteristics of patient records without violating confidentialities of individuals. To generate such datasets over a large set of categorical variables, we propose the use of the maximum entropy principle to formalize a generative model such that in a statistically well-founded way we can optimally utilize given prior information about the data, and are unbiased otherwise. An efficient inference algorithm is designed to estimate the maximum entropy model, and we demonstrate how our approach is adept at estimating underlying data distributions. We evaluate this approach against both simulated data and US census datasets, and demonstrate its feasibility using an epidemic simulation application.

CCS Concepts: •**Computing methodologies** →**Maximum entropy modeling;** •**Information systems** →*Data mining;*

Additional Key Words and Phrases: Multivariate Categorical Data, Synthetic Population, Maximum Entropy Models, Probabilistic Modeling.

ACM Reference format:
Hao Wu, Yue Ning, Prithwish Chakraborty, Jilles Vreeken, Nikolaj Tatti, and Naren Ramakrishnan. 2010. Generating Realistic Synthetic Population Datasets. *ACM Trans. Knowl. Discov. Data.* 9, 4, Article 39 (March 2010), 22 pages.
DOI: 0000001.0000001

1 INTRODUCTION

Many research areas, e.g., epidemiology, public health, social science, study the behavior of large populations of individuals under natural scenarios as well as under human interventions. A key

Author's addresses: H. Wu, Google Inc, Mountain View, CA 94043, USA; Y. Ning and N. Ramakrishnan, Discovery Analytics Center, Virginia Tech, Arlington, VA 22203, USA; P. Chakraborty, IBM Watson Health, Cambridge, MA 02142, USA; J. Vreeken, Max Planck Institute for Informatics and Saarland University, Saarland Informatics Campus, 66123 Saarbrücken, Germany; N. Tatti, Aalto University School of Science, Department of Information and Computer Science, P.O. Box 15400, FI-00076 Aalto, Finland.

DOI: 0000001.0000001

need across these domains is the ready availability of realistic synthetic datasets that can capture key attributes and activities of large populations.

For instance, in epidemiology, synthetic populations are necessary to study disease propagation and intervention measures before implementation. Information from the US census is typically used to model such synthetic datasets. In social science, synthetic populations are necessary to understand how policy decisions might affect preferences and behaviors of individuals. Finally, in public health, synthetic populations are necessary to capture diagnostic and procedural characteristics of patient records without violating confidentialities of individuals.

Typically, the constraints underlying synthetic population generation are assumptions on the supporting marginal or conditional distributions. Although there exist prior studies in estimating probability distributions subject to constraints (e.g., Monte Carlo methods), they are primarily focused on continuous-valued data. Many domains on the other hand, such as those studied here, feature the need for multi-dimensional categorical datasets.

As a case in point, in epidemiology, one important task is to simulate disease spread and potential outbreaks on the city- or nation-level, and provide useful information to public health officials to support policy and decision making. To make such simulations as accurate as possible, synthetic populations that have the same structural and behavioral properties as the real population are needed. In domains like health care, privacy is an additional issue motivating the design of synthetic populations. In these applications, the necessary datasets to be generated can be represented as tuples with categorical data attributes.

Motivated by these emerging needs, we focus our attention on constructing a generative model that captures given characteristics of categorical population attributes, and best estimates the underlying data generation distribution. However, modeling multi-dimensional categorical data and estimating distributions can be quite challenging due to the exponential possibilities of data spaces in terms of the number of dimensions of categorical data tuples. Although many dimension reduction techniques [29] and pattern recognition algorithms [12, 28] have been proposed and studied in many other machine learning and data mining research areas, it is difficult to simply apply them here in our problem setting. To address these challenges and difficulties, we take the first step here to study this problem. To model categorical data with statistical constraints, we apply the classical and statistically well-founded maximum entropy model. We construct a generative maximum entropy model, which takes the data schema (the set of all categorical attributes that appear in the data) and a set of constraint categorical patterns (a subset of categorical attributes with corresponding probabilities of appearance in the data), as shown in Fig. 1, wherein the probabilities of certain categorical patterns are required to satisfy given constraints. In this way, the maximum entropy model maintains the selected characteristics of the underlying categorical data distribution. By sampling the categorical tuples from the maximum entropy model, synthetic population datasets can be generated as illustrated by Fig. 1.

Generally, solving maximum entropy models can be infeasible in practice. In this paper, we show that by leveraging the structure of the categorical data space in our setting, the maximum entropy model could be inferred quite efficiently. We also propose a heuristic together with the Bayesian information criterion (BIC) to select a simple as well as informative model. To summarize our approach in a nutshell, our contributions are:

(1) We formalize the problem of generating synthetic population datasets via a generative maximum entropy model for categorical data, which captures the statistical features of the underlying categorical data distributions.

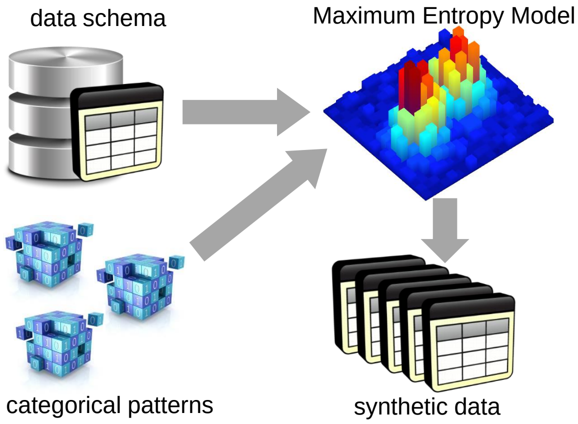

Fig. 1. Process of generating realistic synthetic data with our proposed approach.

(2) By exploring the structure of the categorical data space, we propose a partition scheme to make the maximum entropy model inference more efficient than the general case. We also present an efficient graph-based model inference algorithm.
(3) We propose a BIC-based heuristic to perform model selection wherein the simple and informative maximum entropy model will be chosen.
(4) Using results on both synthetic datasets and real US census data, we demonstrate that the proposed maximum entropy model is capable of recovering the underlying categorical data distribution and generating relevant synthetic populations.

2 PRELIMINARIES

Let $\mathcal{A} = \{A_1, A_2, \ldots, A_q\}$ denote a set of categorical random variables (or attributes), and $\mathcal{R}(A_i) = \{a_1^{(i)}, a_2^{(i)}, \ldots, a_{k_i}^{(i)}\}$ represent the set of k_i possible values for random variable A_i. Here, $|\cdot|$, e.g. $|\mathcal{R}(A_i)|$, is used to represent the cardinality of a set.

By a random categorical tuple, we mean a vector of categorical random variables, e.g. $T = (A_1, A_2, \ldots, A_q)$, which is generated by some unknown probability distribution. The notation of $T(A_i)$ is used to represent the value of attribute A_i in tuple T. The space of all the possible categorical tuples is denoted by $\mathcal{S} = \mathop{\Large\times}_{i=1}^{q} \mathcal{R}(A_i)$, where $\mathop{\Large\times} \cdot$ represents a series of Cartesian product over the given sets. Given a categorical pattern, which is defined as an ordered set $X = (A_i \mid A_i \in C, C \subseteq \mathcal{A})$ over a subset of random variables $C \subseteq \mathcal{A}$, let $\mathcal{S}_X = \mathop{\Large\times}_{A_i \in C} \mathcal{R}(A_i)$ represent the space that contains all the possible values of pattern X. An instantiation of pattern X is defined as $\boldsymbol{x} = \left(a_j^{(i)} \mid a_j^{(i)} \in \mathcal{R}(A_i), A_i \in C, C \subseteq \mathcal{A}\right)$, and $X(A_i)$ is used to represent the value of attribute A_i in the pattern X.

For any pattern value $\boldsymbol{x}$ associated with pattern X, we use the notation of $T = \boldsymbol{x}$ if the corresponding random variables in T equal to the values in $\boldsymbol{x}$ and $p(T = \boldsymbol{x})$ to denote the probability of $T = \boldsymbol{x}$. Given a categorical dataset D, $\tilde{p}(T = \boldsymbol{x} \mid D)$ is used to denote the empirical probability of $T = \boldsymbol{x}$ in the dataset D. An indicator function $I_X(T = \boldsymbol{x}) : \mathcal{S} \longrightarrow \{0, 1\}$ of pattern X, which maps a categorical tuple to a binary value, is defined as:

$$I_X(T = \boldsymbol{x}) = \begin{cases} 1, & \text{if } T = \boldsymbol{x}, \\ 0, & \text{otherwise.} \end{cases}$$

Table 1. Summary of frequently used notation.

A_i	categorical random variable	$\mathcal{A}$	a set of categorical random variable
$a^{(i)}$	value for random variable A_i	$\mathcal{R}(A_i)$	set of possible values for A_i
$\lvert \cdot \rvert$	set cardinality	T	random categorical tuple
$T(A_i)$	value of random variable A_i in T	X	categorical pattern
$X(A_i)$	value of A_i in pattern X	$\boldsymbol{x}$	value instantiation of pattern X
$\mathcal{S}$	entire categorical tuple space	$\mathcal{S}_X$	categorical tuple space spanned by pattern X
p	probability distribution	$\tilde{p}$	empirical probability distribution
D	categorical dataset	$H(p)$	entropy of probability distribution p
$I_X(\cdot)$	indicator function of pattern X	u, v	maximum entropy model parameters

Given a probability distribution p over the categorical tuple space $\mathcal{S}$, the entropy $H(p)$ with respect to p is defined as:

$$H(p) = -\sum_{T \in \mathcal{S}} p(T) \log p(T) \ .$$

The maximum entropy principle states that among a set of probability distributions $\mathcal{P}$ that comply with the given prior information about the data, the maximum entropy distribution

$$p^* = \operatorname*{argmax}_{p \in \mathcal{P}} H(p)$$

will optimally use the current prior information and best summarize the data. Otherwise, it is fully unbiased. Table 1 summarizes the frequently used notations in this paper.

Problem Statement. Given a set of categorical patterns $\mathcal{X}$ with associated empirical probabilities as the prior information of a dataset, find a probabilistic model p that best utilizes the given prior information and helps regenerate categorical datasets that conform to the given prior information.

3 CATEGORICAL MAXIMUM ENTROPY MODEL

3.1 Model Specification

Suppose we have a set of categorical patterns $\mathcal{X} = \{X_i \mid i = 1, 2, \ldots, n\}$ and an associated set of empirical probabilities $\tilde{P} = \{\tilde{p}(T = \boldsymbol{x}_{i,j} \mid D) \mid \boldsymbol{x}_{i,j} \in \mathcal{S}_{X_i}, i = 1, 2, \ldots, n\}$ as prior information about dataset D. Here, $\boldsymbol{x}_{i,j}$ denotes the j^{th} value of the pattern X_i. Notice that it is not necessary that every possible value of pattern X_i in $\mathcal{S}_{X_i}$ is provided as part of the prior information here. Such prior information identifies a group of probability distributions $\mathcal{P}$ over $\mathcal{S}$ which agree with the empirical probabilities of the given categorical patterns. That is:

$$\mathcal{P} = \left\{p \mid p(T = \boldsymbol{x}_{i,j}) = \tilde{p}(T = \boldsymbol{x}_{i,j} \mid D), \forall X_i \in \mathcal{X}, \tilde{p}(T = \boldsymbol{x}_{i,j} \mid D) \in \tilde{P}\right\}. \tag{1}$$

Following the maximum entropy principle [5], for all $p \in \mathcal{P}$, we are particularly interested in the maximum entropy distribution which optimally represents the given prior information. The famous theorem proved by Csiszár [5] (Theorem 3.1) shows that the maximum entropy distribution has an exponential form. In our categorical scenario, the maximum entropy distribution could be written as

$$p^*(T) = u_0 \prod_{X_i \in \mathcal{X}} \prod_{\boldsymbol{x}_{i,j} \in \mathcal{S}_{X_i}} \left(u_{i,j}\right)^{I_{X_i}(T = \boldsymbol{x}_{i,j})} , \tag{2}$$

where $u_{i,j} \in \mathbb{R}$ are the model parameters associated with each model constraint specified in Equation (1), and u_0 is the normalizing constant.

ALGORITHM 1: Sampling random categorical tuples

```
input  : Maximum Entropy model $p^*$.
output : A random categorical tuple $T$.
1 $T \leftarrow \varnothing$;
2 foreach $A_i \in \mathcal{A}$ do
3     foreach $a_j^{(i)} \in \mathcal{R}(A_i)$ do
4         compute the conditional probability $p^*(a_j^{(i)} \mid T)$;
5     $T(A_i) \leftarrow \mathtt{Sample}(\mathcal{R}(A_i), \{p^*(a_j^{(i)} \mid T)\})$;
6 return $T$;
```

3.2 Incorporating Individual Attribute Frequencies

The frequencies of individual attributes play an important role in the pattern analysis and discovery. Such frequencies characterize the attribute marginal distributions which convey basic information about the data currently under investigation, and yet are relatively easy to calculate from the data. Incorporating such individual attribute frequencies will enrich the categorical maximum entropy model, and make it more informative.

Although such individual attribute frequencies can be treated as part of the categorical pattern set $\mathcal{X}$, considering the computation efficiency which will be explained in detail in the next section, the categorical maximum entropy model treats them separately. Let $v_{i,j}$ denote the model parameters corresponding to the individual attribute constraints, then, the maximum entropy distribution can be factorized as:

$$p^*(T) = u_0 \prod_{X_i \in \mathcal{X}} \prod_{\boldsymbol{x}_{i,j} \in \mathcal{S}_{X_i}} \left(u_{i,j}\right)^{I_{X_i}(T=\boldsymbol{x}_{i,j})} \times \prod_{A_i \in \mathcal{A}} \prod_{a_j \in \mathcal{R}(A_i)} \left(v_{i,j}\right)^{I_{A_i}(T=a_j)} . \tag{3}$$

Notice that the second component involved with $v_{i,j}$ also follows the exponential form described in Equation (2). By introducing a normalizing constant v_0, an independent maximum entropy distribution $p_{\mathcal{A}}(T)$ that only involves individual attribute constraints could be defined as:

$$p_{\mathcal{A}}(T) = v_0 \prod_{A_i \in \mathcal{A}} \prod_{a_j \in \mathcal{R}(A_i)} \left(v_{i,j}\right)^{I_{A_i}(T=a_j)} . \tag{4}$$

Combining Equation (3) and (4), the maximum entropy distribution that incorporates individual attribute frequencies would be specified as:

$$p^*(T) = p_{\mathcal{A}}(T) \frac{u_0}{v_0} \prod_{X_i \in \mathcal{X}} \prod_{\boldsymbol{x}_{i,j} \in \mathcal{S}_{X_i}} \left(u_{i,j}\right)^{I_{X_i}(T=\boldsymbol{x}_{i,j})} . \tag{5}$$

3.3 Sampling Categorical Tuples from the Model

The ideal way to sample categorical tuples from the model would be first computing the probabilities for each tuple $T \in \mathcal{S}$ under the maximum entropy distribution p^*, and then sampling the tuples from the complete tuple space $\mathcal{S}$ based on their probabilities. However, such straightforward approach is not feasible in practice when $\mathcal{S}$ is large enough.

Instead, we propose an attribute-wise sampling approach. Algorithm 1 summarizes the procedure that generates random categorical tuples from the maximum entropy model p^*. To sample random tuples, we take the following steps. Starting with a empty categorical tuple T (line 1), for each categorical attribute $A_i \in \mathcal{A}$, we compute the probability of each possible value $a_j^{(i)} \in \mathcal{R}(A_i)$ conditioned on those attribute values we have already sampled in tuple T (line 3 – 4). Based on

ALGORITHM 2: Iterative Scaling for Categorical Maximum Entropy Model

```
input  : A set of categorical patterns $\mathcal{X}$, and associated empirical probabilities $\tilde{P}$.
output : The maximum entropy model $p^*$.
1 Initialize $p(T) = \frac{1}{|\mathcal{S}|}, \forall T \in \mathcal{S}$;
2 while $p$ is not converged do
3     foreach $X_i \in \mathcal{X}$ do
4         foreach $\boldsymbol{x}_{i,j} \in \mathcal{S}_{X_i}$ s.t. $\tilde{p}(T = \boldsymbol{x}_{i,j}) \in \tilde{P}$ do
5             Compute $p(T = \boldsymbol{x}_{i,j})$;
6             $u_{i,j} \leftarrow u_{i,j} \cdot \frac{\tilde{p}(T=\boldsymbol{x}_{i,j}|D)}{p(T=\boldsymbol{x}_{i,j})} \cdot \frac{1-p(T=\boldsymbol{x}_{i,j})}{1-\tilde{p}(T=\boldsymbol{x}_{i,j}|D)}$;
7             $u_0 \leftarrow u_0 \cdot \frac{1-\tilde{p}(T=\boldsymbol{x}_{i,j}|D)}{1-p(T=\boldsymbol{x}_{i,j})}$;
8 return $p$;
```

the conditional probability distribution just calculated for attribute A_i, the value of A_i in tuple T is randomly sampled from its range $\mathcal{R}(A_i)$ (line 5). Notice that the order of selecting attributes could be random, and from statistical point of view, this should not affect the sampling result.

4 MODEL INFERENCE

In this section, we develop an efficient algorithm to infer the categorical maximum entropy model. To infer the categorical maximum entropy model, we need to find the values of model parameters u_0 and $u_{i,j}$ (also v_0 and $v_{i,j}$ if individual attribute constraints are involved). Our algorithm is built on the well-known Iterative Scaling [6] framework, which is described in Algorithm 2. The general idea of the algorithm is that starting from the uniform distribution, it iteratively updates each model parameter to make the distribution satisfy the corresponding constraint until it converges to the maximum entropy distribution. The proof of convergence for the Iterative Scaling algorithm is out of the scope for this paper. Readers who are interested in the proof of convergence can refer to the paper by Darroch and Ratcliff [6] for details. A crucial step in the Iterative Scaling algorithm is to compute the probability of every categorical pattern $X_i = \boldsymbol{x}_{i,j} \in \mathcal{X}$ under the current estimation of maximum entropy distribution p, which could be simply calculated as $p(T = \boldsymbol{x}_{i,j}) = \sum p(T)$ where $I_{X_i}(T = \boldsymbol{x}_{i,j}) = 1$. However, such straightforward strategy is infeasible in our problem setting since it will result a computational complexity of $\prod_{A_i \in \mathcal{A}} |\mathcal{R}(A_i)|$ for a single model parameter update. In fact, querying maximum entropy models has been shown to be **PP**-hard [30]. To overcome such challenge, we present our proposed efficient model inference algorithm in the rest of this section.

4.1 Efficient Model Inference

In order to efficiently query the maximum entropy model during the iterative updates of the model parameters, we need to explore the particular structure of the tuple space $\mathcal{S}$ determined by the given pattern set $\mathcal{X}$. We will start with the simpler case where individual attribute constraints are not involved. After examining the exponential form of the maximum entropy distribution in Equation (2), we observe that for any two categorical tuples T_1 and T_2 in $\mathcal{S}$, if they contain the same subset of categorical patterns in $\mathcal{X}$, they will have the same probability under the maximum entropy distribution inferred based on $\mathcal{X}$. In other words, $\forall T_1, T_2 \in \mathcal{S}$, if $I_{X_i}(T_1 = \boldsymbol{x}_{i,j}) = I_{X_i}(T_2 = \boldsymbol{x}_{i,j})$ holds true for all $X_i \in \mathcal{X}$ and $\tilde{p}(T = \boldsymbol{x}_{i,j} \mid D) \in \tilde{P}$, then $p^*(T_1) = p^*(T_2)$. Based on such observation, we have the following definition of tuple block.

Definition 4.1. A tuple block B is a set categorical tuples such that $\forall T_1, T_2 \in B$, $I_{X_i}(T_1 = \boldsymbol{x}_{i,j}) = I_{X_i}(T_2 = \boldsymbol{x}_{i,j})$ holds true for all $X_i \in \mathcal{X}$, $\boldsymbol{x}_{i,j} \in \mathcal{S}_{X_i}$, *and* $\tilde{p}(T = \boldsymbol{x}_{i,j} \mid D) \in \tilde{P}$.

ALGORITHM 3: Constructing the tuple block graph

```
input  : A set of categorical patterns $\mathcal{X}$, and associated empirical probabilities $\tilde{P}$.
output : tuple block graph $G$.
Let $G \leftarrow \{\varnothing\}$;
foreach $X_i \in \mathcal{X}, \boldsymbol{x}_{i,j} \in \mathcal{S}_{X_i}$ $s.t.$ $\tilde{p}(T = \boldsymbol{x}_{i,j}) \in \tilde{P}$ do
    foreach $B_k \in G$ do
        $B_{new} \leftarrow$ createBlock($B_k$, $X_i$);
        if $B_{new} \neq Null$ then
            findPosition($\varnothing$, $Null$, $B_{new}$);
return $G$;
```

With the definition of tuple block, we could partition the entire categorical tuple space into several tuple blocks. When $|\mathcal{X}| \ll |\mathcal{A}|$, the partition scheme introduced here could greatly reduce the dimensionality of the space we are working on. Here, we use $\mathcal{B}_{\mathcal{X}}$ to denote the tuple block space generated based on pattern set $\mathcal{X}$. Also, the definition of tuple block let us extend the indicator function defined over tuple space to the domain of tuple block, which is defined as:

$$I_{X_i}(B \mid \boldsymbol{x}_{i,j}) = I_{X_i}(T = \boldsymbol{x}_{i,j}), \quad \forall X_i \in \mathcal{X}, T \in B.$$

By introducing tuple blocks, we transfer the problem of computing categorical pattern probability $p(T = \boldsymbol{x}_{i,j})$ on tuple space to the block space, which makes it possible to calculate $p(T = \boldsymbol{x}_{i,j})$ in a reasonable time. In the context of tuple blocks, the pattern probability $p(T = \boldsymbol{x}_{i,j})$ would be

$$p(T = \boldsymbol{x}_{i,j}) = \sum_{\substack{B \in \mathcal{B}_{\mathcal{X}}, \\ I_{X_i}(B|\boldsymbol{x}_{i,j})=1}} p(B)\,,$$

where $p(B)$ is the probability for tuple block B. Since the probabilities for the categorical tuples within the same block are all the same, the probability for the tuple block B is defined as:

$$p(B) = \sum_{T \in B} p(T) = |B| \times u_0 \prod_{X_i \in \mathcal{X}} \prod_{\boldsymbol{x}_{i,j} \in \mathcal{S}_{X_i}} (u_{i,j})^{I_{X_i}(B|\boldsymbol{x}_{i,j})}\,.$$

Now, our problem comes down to how to organize the tuple block space $\mathcal{B}_{\mathcal{X}}$ and efficiently compute the number of categorical tuples in each block, or in other words, the size $|B|$ of each tuple block B. In order to achieve that, we introduce a partial order on $\mathcal{B}_{\mathcal{X}}$. Let

$$attr(B) = \bigcup_{\substack{X_i \in \mathcal{X}, \\ I_{X_i}(B|\boldsymbol{x}_{i,j})=1}} X_i\,,$$

which represents the set of attributes involved by the categorical patterns that tuple block B contain. Then, we have the definition about the partial order over $\mathcal{B}_{\mathcal{X}}$ as described below.

Definition 4.2. Given any tuple blocks $B_1, B_2 \in \mathcal{B}_{\mathcal{X}}$, $B_1 \subseteq B_2$ if and only if the following conditions hold true:

(1) $attr(B_1) \subseteq attr(B_2)$;
(2) $B_1(A_k) = B_2(A_k),\ \forall A_k \in attr(B_1) \cap attr(B_2)$.

Here, $B(A_k)$ denotes the value of attribute A_k in the tuple block B. It is easy to verify that Definition 4.2 satisfies the property of reflexivity, antisymmetry and transitivity.

With the partial order $\subseteq$ defined on $\mathcal{B}_{\mathcal{X}}$ here, it is natural to organize the tuple blocks into a hierarchical graph structure. That is, if tuple block $B_k \subseteq B_l$, block B_l is organized as the child

ALGORITHM 4: *findPosition* procedure

```
input  : Current block $B_{curr}$, last visited block $B_{last}$, new block $B_{new}$.
output : Success or Fail.
if $B_{new}$ and $B_{curr}$ are the same then
    return Success;
else if $B_{new} \subseteq B_{curr}$ then
    $child(B_{last}) \leftarrow child(B_{last}) \setminus \{B_{curr}\}$;
    $child(B_{new}) \leftarrow child(B_{new}) \cup \{B_{curr}\}$;
    $child(B_{last}) \leftarrow child(B_{last}) \cup \{B_{new}\}$;
    return Success;
else if $B_{curr} \subseteq B_{new}$ then
    if $child(B_{curr}) = \emptyset$ then
        $child(B_{curr}) \leftarrow child(B_{curr}) \cup \{B_{new}\}$;
        return Success;
    else
        $failBlock \leftarrow$ InsertDescendant($B_{new}, B_{curr}$);
        checkDescendant($failBlock, B_{new}$);
        return Success;
return Fail;
Procedure InsertDescendant($B_{new}, B_{curr}$):
    $failBlock \leftarrow \emptyset$, $accu \leftarrow$ Fail;
    foreach $B_k \in child(B_{curr})$ do
        $r \leftarrow$ findPosition($B_k, B_{curr}, B_{new}$);
        if $r = Success$ then
            $accu \leftarrow$ Success;
        else
            $failBlock \leftarrow failBlock \cup \{B_k\}$;
    if $accu = Fail$ then
        $child(B_{curr}) \leftarrow child(B_{curr}) \cup \{B_{new}\}$;
    return $failBlock$;
```

of block B_k. Algorithm 3 illustrates how such block graph is constructed and maintained. The algorithm starts with the graph that has only one block represented by $\varnothing$ indicating that none of the categorical patterns is involved in this block (line 1). We will refer this block as root block in the rest of this section. Then, for each of the categorical pattern $X_i \in \mathcal{X}$ and its possible value $\boldsymbol{x}_{i,j}$, we attempt to create a new tuple block by merging it with every existing block B_k from root level to leaf level (without child blocks) in the current block graph G if they are compatible (line 4). A categorical pattern X_i is not compatible with tuple block B_k if $attr(B_k) \cap X_i \neq \emptyset$, and $\exists A_i \in attr(B_k) \cap X_i$ such that $B_k(A_i) \neq X_i(A_i)$. If a new tuple block B_{new} is created, it is obvious that for all $X_l \in \mathcal{X}, I_{X_l}(B_k \mid \boldsymbol{x}_{l,j}) = 1$, we have $I_{X_l}(B_{new} \mid \boldsymbol{x}_{l,j}) = 1$ and also $I_{X_i}(B_{new} \mid \boldsymbol{x}_{i,j}) = 1$. Finally, the new tuple block B_{new} will be added into the current block graph G based on the partial order described in Definition 4.2 (line 6).

To be more specific, Algorithm 4 illustrates how the procedure *findPosition* inserts a new tuple block into the block graph G in a recursive manner. Depending on the relationship between the current block B_{curr} we are visiting and the new block B_{new}, the insertion operation could be classified into four scenarios.

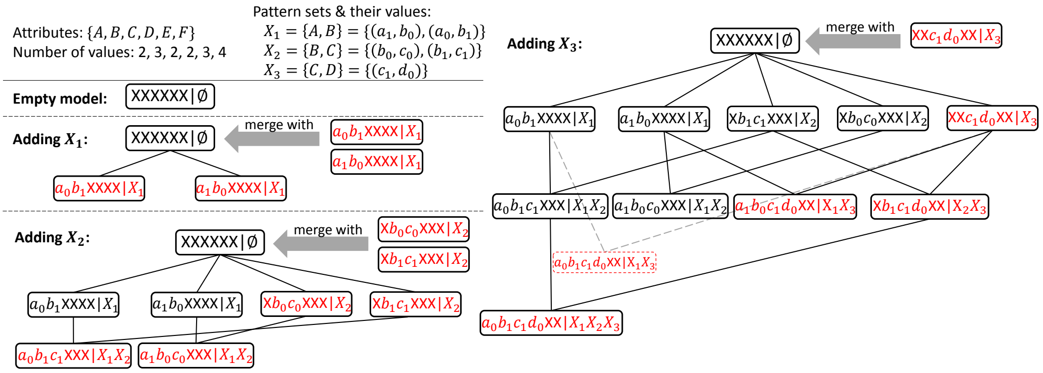

Fig. 2. Example of constructing the tuple block graph on a toy dataset with 6 attributes and 3 categorical patterns. The blocks marked with red denote the new tuple blocks created in each iteration by adding new categorical patterns.

Case 1: B_{new} and B_{curr} are the same tuple block. Two tuple block B_k and B_l are considered to be the same if they cover the same set of categorical patterns, e.g. $\forall X_i \in \mathcal{X}, \boldsymbol{x}_{i,j} \in \mathcal{S}_{X_i}\ s.t.\ \tilde{p}(T = \boldsymbol{x}_{i,j}) \in \tilde{P}$, we have $I_{X_i}(B_k \mid \boldsymbol{x}_{i,j}) = I_{X_i}(B_l \mid \boldsymbol{x}_{i,j})$. Since block B_{new} and B_{curr} are the same and B_{curr} is already part of the block graph, inserting B_{new} into block graph is not necessary any more. Thus, we simply return *Success* in this scenario (line 1 – 2).

Case 2: $B_{new} \subseteq B_{curr}$. In this case, the new tuple block B_{new} should be inserted between block B_{last} and B_{curr}, where B_{last} is the last visited tuple block. To achieve this, block B_{curr} is first removed from the child block set of B_{last}, and added as a child block of B_{new}. Finally, the new block B_{new} is inserted as a child block of B_{last}, and *Success* is returned (line 3 – 7).

Case 3: $B_{curr} \subseteq B_{new}$. In this scenario, the new tuple block B_{new} should be inserted as a descendant of the current block B_{curr}. Depending on whether the block B_{curr} has any child blocks, the insertion operation can be further divided into two sub-cases:

- **Case 3.1:** block B_{curr} has no child block. In this scenario, the new block B_{new} is directly inserted as a new child of B_{curr} (line 9 – 11);
- **Case 3.2:** block B_{curr} has child blocks. Then, for each child block of B_{curr}, the *findPosition* procedure is recursively performed to find the correct position to insert block B_{new} (line 19 – 24). If none of these operations succeeds, block B_{new} will be inserted as a new child block of B_{curr} (line 25 – 26). At last, the descendants of the child blocks of B_{curr} on which the *findPosition* procedure failed to insert the block B_{new} are further examined to see whether any of them could satisfy the partial order with block B_{new} and be added as a child block of B_{new} (line 14, *checkDescendant* procedure).

Case 4: B_{new} does not have any particular relationship with B_{curr}. In this case, nothing needs to done with the tuple blocks B_{curr} and B_{new}, and *Fail* is simply returned to indicate that the attempt to insert block B_{new} is failed.

Figure 2 shows an example of constructing such hierarchical block graph on a small toy dataset with 6 attributes and 3 categorical patterns. With the block graph G, the size of the tuple block could be easily calculated using the set inclusion-exclusion principle. We first define the cumulative size of a tuple block B, which is given by

$$cum(B) = \prod_{A_i \in \mathcal{A} \setminus attr(B)} |\mathcal{R}(A_i)| \ .$$

ALGORITHM 5: *computeBlockSize* procedure

```
input  : tuple block graph $G$, current visited block $B_{curr}$.
output : Block size for each $B \in \mathcal{B}_{\mathcal{X}}$.
$cum(B_{curr}) \leftarrow \prod_{A_i \in \mathcal{A} \setminus attr(B_{curr})} |\mathcal{R}(A_i)|$;
if $child(B_{curr}) = \emptyset$ then
    $|B_{curr}| \leftarrow cum(B_{curr})$;
    return;
foreach $B_k \in child(B_{curr})$ do
    computeBlockSize($G, B_k$);
$|B_{curr}| \leftarrow cum(B_{curr}) - \sum_{B_k \in desc(B_{curr})} |B_k|$;
return;
```

Then the actual block size for block B could be computed as

$$|B| = cum(B) - \sum_{B_k \in \mathcal{B}_{\mathcal{X}}, B \subseteq B_k} |B_k| \ .$$

In the block graph G, the tuple blocks that satisfy $B_k \in \mathcal{B}_{\mathcal{X}}, B \subseteq B_k$ are simply those descendant blocks of B. Algorithm 5 describes the procedure of computing block size for each tuple block in $\mathcal{B}_{\mathcal{X}}$ with the block graph G, where $desc(B)$ represents the set of descendant blocks of B in the graph G.

When individual attribute constraints are taken into account, the problem becomes a little more complicated. However, it is obviously not feasible to combine the individual attribute constraints with the categorical pattern constraints together and construct the tuple block graph. This will make the tuple block space blow up. Instead, as we mentioned previously in Section 3, the individual attribute constraints are modeled with a separate maximum entropy distribution $p_{\mathcal{A}}$, defined in Equation (4), which only considers these constraints. The block graph G is still constructed based on the categorical patterns in $\mathcal{X}$, which will exactly have the same structure as before. In this case, following the same logic, the probability for tuple block B becomes

$$p(B) = p_{\mathcal{A}}(B) \cdot \frac{u_0}{v_0} \cdot \prod_{X_i \in \mathcal{X}} \prod_{\mathbf{x}_{i,j} \in \mathcal{S}_{X_i}} (u_{i,j})^{I_{X_i}(B|\mathbf{x}_{i,j})} \ ,$$

where $p_{\mathcal{A}}(B) = \sum_{T \in B} p_{\mathcal{A}}(T)$ denotes the probability of tuple block B under the separate maximum entropy distribution $p_{\mathcal{A}}$. Thus, the problem of computing the probability $p(T = \mathbf{x}_{i,j})$ becomes calculating probabilities of tuple blocks $p_{\mathcal{A}}(B)$ for each $B \in \mathcal{B}_{\mathcal{X}}$. Since $p_{\mathcal{A}}$ only takes the individual attribute constraints into account, every attribute is independent of each other under the maximum entropy distribution $p_{\mathcal{A}}$. Similar to the cumulative size of a tuple block, we define the cumulative probability of a tuple block under $p_{\mathcal{A}}$ as

$$p_{\mathcal{A}}^{(c)}(B) = \prod_{A_i \in attr(B)} p_{\mathcal{A}}\left(T = a_j^{(i)}\right) \ ,$$

where $a_j^{(i)}$ is the value of attribute A_i associated with tuple block B. With the exponential form described in Equation (4), it is not difficult to verify that the probability of $T = a_j^{(i)}$ under maximum entropy distribution $p_{\mathcal{A}}$ is:

$$p_{\mathcal{A}}\left(T = a_j^{(i)}\right) = \frac{v_{i,j}}{\sum_{l=1}^{k_i} v_{i,l}} \ .$$

Again, to compute $p_{\mathcal{A}}(B)$ for all $B \in \mathcal{B}_{\mathcal{X}}$ with the set inclusion-exclusion principle, we could directly apply the *computeBlockSize* procedure with $|B|$ and $cum(B)$ replaced by $p_{\mathcal{A}}(B)$ and $p_{\mathcal{A}}^{(c)}(B)$ respectively.

Notice that the model parameters $v_{i,j}$ also need to be updated in the Iterative Scaling framework. However, the block graph G is constructed without considering individual attribute patterns, which makes it difficult to compute the probabilities of these individual attribute patterns under the maximum entropy model directly from the block graph G. In order to get these probabilities, we treat these individual attribute patterns as arbitrary categorical patterns and query their probabilities from the maximum entropy model. The detail of querying the maximum entropy model will be described in the next section.

Finally, the model inference algorithm could be further optimized in the following way. Suppose the categorical patterns in $\mathcal{X}$ could be divided into two disjoint groups, e.g. $\mathcal{X}_1, \mathcal{X}_2 \subset \mathcal{X}$ and $\mathcal{X}_1 \cup \mathcal{X}_2 = \mathcal{X}$ such that $\forall X_1 \in \mathcal{X}_1, \forall X_2 \in \mathcal{X}_2$ we have $X_1 \cap X_2 = \emptyset$. In this case, the maximum entropy model $p^*_{\mathcal{X}}$ over $\mathcal{X}$ could be factorized into two independent components $p^*_{\mathcal{X}_1}$ and $p^*_{\mathcal{X}_2}$ such that $p^*_{\mathcal{X}} = p^*_{\mathcal{X}_1} \cdot p^*_{\mathcal{X}_2}$. Furthermore, $p^*_{\mathcal{X}_1}$ and $p^*_{\mathcal{X}_2}$ only rely on pattern set $\mathcal{X}_1$ and $\mathcal{X}_2$, respectively. Such decomposition greatly reduces the sizes of tuple block spaces $\mathcal{B}_{\mathcal{X}_1}$ and $\mathcal{B}_{\mathcal{X}_2}$ compared to the original $\mathcal{B}_{\mathcal{X}}$, and could also be extended to the scenario when there are multiple such disjoint pattern groups. Due to the independence between these maximum entropy components, they can also be inferred parallelly to further speed up the model inference process.

4.2 Querying the Model

Given an arbitrary categorical pattern $X' \notin \mathcal{X}$ with associated value $\boldsymbol{x}'$, to query its probability under the maximum entropy distribution p^*, we perform the following operations. Let $\mathcal{X}' = \mathcal{X} \cup \{X'\}$, and a temporary tuple block graph G' is constructed by applying the procedure described in Algorithm 3 over categorical pattern set $\mathcal{X}'$. Then the size of each tuple block in graph G' is computed by calling *computeBlockSize* procedure, and the probability of categorical pattern X' is given by

$$p^*(T = \boldsymbol{x}') = \sum_{\substack{B \in \mathcal{B}_{\mathcal{X}'} \\ I_{\mathcal{X}'}(B|\boldsymbol{x}')=1}} p^*(B) \,.$$

4.3 Computational Complexity

Constructing the tuple block graph (Algorithm 3) requires $|\mathcal{B}_{\mathcal{X}}|$ insertion operations. Since the block graph has a hierarchical structure, let's define the depth of the tuple block graph as the maximum number of hops (parent-child relationship) from the root block $\varnothing$ to the leaf block. Notice that the parent-child relationship between tuple blocks in the graph is based on the partial order described in Definition 4.2, which indicates that the maximum possible depth of the tuple block graph would be $|\mathcal{A}|$. Thus, the complexity of constructing the tuple block graph in the worst case would be $O(|\mathcal{A}| \cdot |\mathcal{B}_{\mathcal{X}}|)$. When inferring the maximum entropy model, the probability of each tuple block needs to be calculated for each model parameter update (Algorithm 5), which results a complexity of $O(|\mathcal{B}_{\mathcal{X}}|)$. If we let N denote the number of model parameters, the complexity of inferring the maximum entropy model would be $O(K \cdot N \cdot |\mathcal{B}_{\mathcal{X}}|)$, where K is the number of iterations required for the proposed inferring algorithm to converge.

5 MODEL SELECTION

In order to discover the most informative prior information from the pattern set $\mathcal{X}$, we adopt the Bayesian Information Criterion (BIC), defined as:

$$BIC_{\mathcal{X}} = -2 \log \mathcal{L}_{\mathcal{X}} + N \cdot \log |D| \,,$$

where $\log \mathcal{L}_{\mathcal{X}}$ denotes the log-likelihood of the maximum entropy model inferred over the pattern set $\mathcal{X}$, N represents the number of model parameters, and $|D|$ is the number categorical tuples in the dataset D. With the exponential form of the maximum entropy distribution specified in Equation (2), its log-likelihood given dataset D is equal to

$$\log \mathcal{L}_{\mathcal{X}} = \sum_{T \in D} \log p^*(T) = |D| \left(\log u_0 + \sum_{X_i \in \mathcal{X}} \sum_{\boldsymbol{x}_{i,j} \in \mathcal{S}_{X_i}} \tilde{p}(T = \boldsymbol{x}_{i,j} \mid D) \cdot \log u_{i,j} \right).$$

The ideal approach to select the most informative categorical patterns from the pattern set $\mathcal{X}$ would be finding a subset of $\mathcal{X}$ that minimizes the BIC score of the model. However, notice that this approach involves a number of model inference operations which is proportional to the number of subsets of $\mathcal{X}$. Considering the computation required for the model inference, this method may be infeasible in practice. Hence, we resort to heuristics. Basically, what we desire are the patterns whose empirical probabilities diverge most from their probabilities under current maximum entropy model. In this case, they will contain the most new information compared to what the model already knows. Thus, we borrow the idea from Kullback-Leibler (KL) divergence, where we make the probability of the categorical pattern X under consideration as one term and the rest of the probability mass as the other term. To be more specific, the heuristic we use is defined as

$$h(\alpha, \beta) = \alpha \log \frac{\alpha}{\beta} + (1 - \alpha) \log \frac{1 - \alpha}{1 - \beta} .$$

Instead of directly searching in the space of power set of $\mathcal{X}$, we adopt an iterative search strategy. Starting from the empty model without any prior information, in each iteration, we choose the pattern $X \in \mathcal{X}$ that maximizes the heuristic $h(p^*(T = \boldsymbol{x}), \tilde{p}(T = \boldsymbol{x} \mid D))$ to update the current maximum entropy model. Here, $p^*(T = \boldsymbol{x})$ and $\tilde{p}(T = \boldsymbol{x} \mid D)$ denote the probability of pattern X under current maximum entropy model and its empirical probability in the given dataset D, respectively. As the model incorporates more and more patterns in $\mathcal{X}$, it becomes more certain about the data, and the negative log-likelihood decreases. However, the model becomes more complicated at the same time, and the penalty term in BIC becomes large. This procedure continues until the BIC score does not decrease any more.

6 EXPERIMENTAL RESULTS

6.1 Synthetic Data Generation

To evaluate the proposed maximum entropy model against the true generating distribution of categorical data, we generate synthetic datasets. Usually when the entire categorical data space is large, it is infeasible to specify an exact generating distribution for categorical data. Thus, we generate the synthetic data D with the following approach.

A set of categorical attributes $\mathcal{A}$ is first generated, and the number of possible values for each attribute $A_i \in \mathcal{A}$ is randomly sampled from a given range. Each categorical attribute A_i is associated with a random generated probability distribution (marginal distribution) that specifies the probability of each possible value of A_i. In order to enforce dependencies between attributes, a set of categorical patterns $\mathcal{X}$ is generated and each of these patterns is associated with a probability. To generate a categorical tuple in the synthetic dataset, we sample from a Bernoulli distribution parameterized by the pattern frequency of each $X \in \mathcal{X}$ to determine whether this tuple should contain this pattern or not. If conflicts occur, the current pattern X will not be added into the tuple. For the rest of the attributes that are not covered by any of these patterns in $\mathcal{X}$, their values in the generated categorical tuple are sampled independently from their corresponding marginal distributions respectively. Such process is repeated to obtain the desired number of categorical

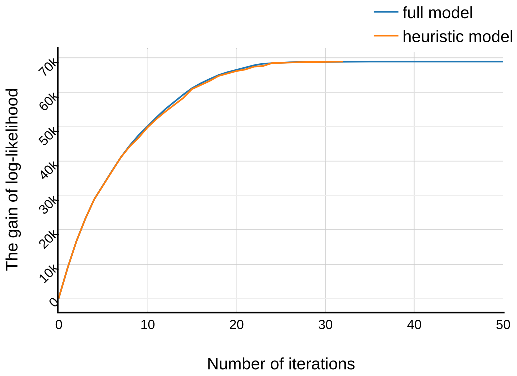

Fig. 3. The gain of the log-likelihood of the full model and heuristic model compared to the base line model. The blue line and orange line are so close that they overlap with each other in some iterations. Also notice that orange line for heuristic model stops early due to the model selection with BIC.

tuples in the synthetic dataset. In our experiments, we set $|\mathcal{A}| = 100$, $|\mathcal{X}| = 50$, and $|D| = 10,000$. All the experiments were conducted on a 80-core Xeon 2.4 GHz machine with 1 TB memory, and the results were averaged across 40 independent runs.

6.2 Results on Synthetic Data

We first verify that the heuristic function $h(\alpha, \beta)$ proposed in Section 5 could discover the most informative patterns from $\mathcal{X}$ based on the current knowledge that the model already knows. We refer the maximum entropy model inferred with entire pattern set $\mathcal{X}$ and all the individual attribute frequencies as *full model*, and the maximum entropy model selected by the heuristic and BIC as *heuristic model*. Notice that in the heuristic model, individual attribute frequencies are also taken into account. In this experiment, we iteratively updated the model with the patterns in $\mathcal{X}$, and measured the log-likelihood in each iteration. However, using BIC to select the model may result different number of patterns incorporated over different synthetic datasets. Thus, we report the results over a single synthetic dataset here. For the full model, the pattern in $\mathcal{X}$ that maximized the log-likelihood in each iteration was selected and added to the model.

Figure 3 illustrates the gain of the log-likelihood as the model incorporates more and more patterns in $\mathcal{X}$. As expected, the gain of the log-likelihood of the full model is larger in some iterations since it identifies the optimal pattern in each iteration with respect to the likelihood. We also observe that although not optimal, the log-likelihood of the heuristic model approximates that of the full model quite well, which demonstrates that the proposed heuristic successfully identifies the relatively informative patterns in each iteration. In the last few iterations, the gain of log-likelihood of the full model barely changes. This indicates that the patterns selected in these iterations are less informative or even redundant.

To assess the quality of the reconstruction, we aim to apply the KL divergence measure. However, in practice, it is very difficult to compute the KL divergence between the entire maximum entropy distribution and data generating distribution for the categorical data due to the large categorical tuple space. As a trade off, we use the probabilities of patterns in pattern set $\mathcal{Y}$ to characterize the probability distributions for categorical data in both scenarios, and define the following approximate

Table 2. Comparison of the approximate KL-divergence measure between full model, heuristic model and baseline model. Standard errors are shown in the parentheses.

		full model	heuristic	baseline
All	$\hat{KL}(p^*, p')$	**0.006013**	0.009344	1.8126
		(**0.01983**)	(0.02516)	(0.3534)
	$\hat{KL}(\tilde{p}, p')$	**0.1767**	0.1840	1.9823
		(**0.02404**)	(0.02805)	(0.3600)
Multi-attribute	$\hat{KL}(p^*, p')$	**0.005871**	0.009204	1.8126
		(**0.01974**)	(0.02501)	(0.3534)
	$\hat{KL}(\tilde{p}, p')$	**0.02837**	0.03322	1.8330
		(**0.01685**)	(0.02339)	(0.3579)

Table 3. Comparison of model preparation time (t_{pre}), model inference time (t_{infer}) and data sampling time (t_{sample}) between full model and heuristic model (in seconds). Standard Errors are shown in the parentheses.

	t_{pre}	t_{infer}	t_{sample}
full model	2750.432	22.553	1.828
	(1512.416)	(14.341)	(1.002)
heuristic model	**15.103**	**11.150**	**0.488**
	(7.844)	(6.549)	(0.234)

KL-divergence measure:

$$\hat{KL}(p^*, p') = \sum_{X \in \mathcal{Y}} \left[p^*(X) \log \frac{p^*(X)}{p'(X)} + (1 - p^*(X)) \log \frac{1 - p^*(X)}{1 - p'(X)} \right].$$

Here, p^* and p' denote the maximum entropy distribution and data generating distribution respectively, and pattern set $\mathcal{Y}$ could be only categorical pattern set $\mathcal{X}$ or $\mathcal{X} \cup \mathcal{A}$ if individual attribute frequencies are considered. We also compute the $\hat{KL}(\tilde{p}, p')$ to compare the empirical probability distribution, say $\tilde{p}$, in the samples generated by the categorical maximum entropy model with the true data generating distribution. In this experiment, we computed $\hat{KL}(p^*, p')$ and $\hat{KL}(\tilde{p}, p')$ for both full model and heuristic model. For comparison purpose, we used independent attribute model $p_{\mathcal{A}}$ where each categorical attribute is independent of each other as the baseline model. For each of these models under consideration, 1000 categorical data samples were generated to compute empirical probability distribution $\tilde{p}$.

Table 2 compares these approximate KL-divergence measures for the scenarios where $\mathcal{Y} = \mathcal{X} \cup \mathcal{A}$ (row *All* in Table 2) and $\mathcal{Y} = \mathcal{X}$ (row *Multi-attribute* in Table 2). In Table 2, the small approximate KL-divergence values for the full model and the heuristic model in the row *All* indicate that the categorical maximum entropy distributions converge to the underlying data generation distribution, and the samples generated by these two models successfully maintain the properties of the data generation distribution. More important, the small approximate KL-divergence values in the row *Multi-attribute* of Table 2 also indicate that the inferred categorical maximum entropy models successfully capture the various multivariate dependencies among multiple categorical attributes. All these results demonstrate that our model is capable of recovering the true categorical data

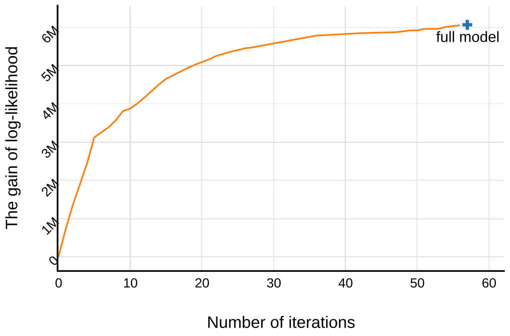

Fig. 4. The gain of the negative log-likelihood of the model compared to the baseline model (model at iteration 0) over the Virginia ACS summary data. The data point marked with cross denotes the negative log-likelihood of the full model where all the categorical patterns in Virginia ACS summary data are considered.

distribution and maintaining its dependency structures. When compared to the baseline model, our model outperforms several magnitudes in term of estimation accuracy.

We also measure the time required to prepare the pattern set that serves as prior information of the model t_{pre}, the time to infer the maximum entropy model t_{infer}, and the time to sample a single categorical tuple from the model t_{sample}. Here, for the full model, t_{pre} refers to the time required to arrange the pattern set $\mathcal{X}$ into the same order used in the iterative model update procedure in the first experiment where the categorical pattern that maximizes the log-likelihood is chosen in each iteration. Table 3 compares the runtime performance between the full model and the heuristic model. With the informative as well as simple model selected by the heuristic function $h(\alpha, \beta)$ and BIC, the heuristic model requires much less time to infer the maximum entropy distribution and sample categorical tuples from the model.

6.3 Results on Real Data

To evaluate the performance of the proposed categorical maximum entropy model on real data, we study the problem of generating synthetic populations with US census data. Specifically, we use the 2012 American Community Survey (ACS) 1-year summary data [33], which contains aggregated statistics about age, sex, race, income, and many other features. Some of these features, e.g. sex and race, are perfect categorical attributes for the proposed maximum entropy model. Although some other features, e.g. age and income, are numerical, they are binned into several ranges based on their values, and treated here as categorical attributes.

In our experiments, we chose the state of Virginia as our study case. Among all the features in the ACS summary data, we selected *sex*, *age*, *race*, *income*, *occupation*, *marital status*, *means of transportation to work*, *education level*, and *health insurance coverage* as the set of categorical attributes. We converted the corresponding aggregated statistics in the ACS summary data into categorical patterns, and inferred the heuristic model over these patterns. Figure 4 describes the gain of the log-likelihood of the heuristic model, and the approximate KL-divergence measure between the inferred maximum entropy distribution and the empirical data distribution in the Virginia ACS summary data is **0.0001975**. Notice that in Figure 4, the last data point marked with a cross indicates the gain of the log-likelihood of the full model where all the categorical patterns

Table 4. Top categorical patterns selected by the heuristic model from the Virginia ACS summary data.

patterns	number of possible values	number of selected values
{means of transportation to work, occupation}	49	34
{sex, income}	8	2
{sex, marital status}	10	2
{sex, age}	8	1

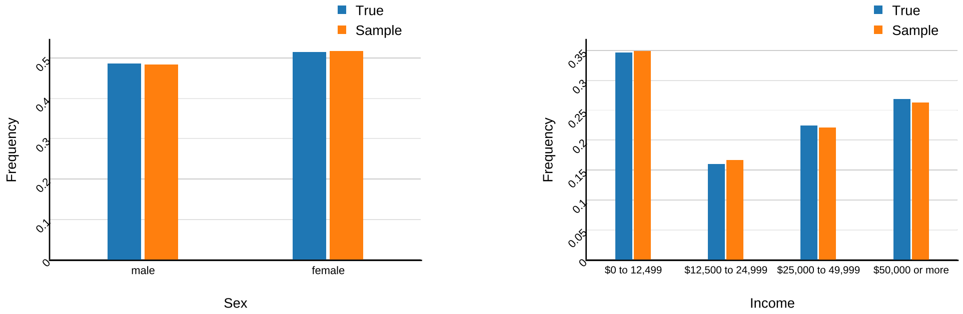

Fig. 5. Comparison of single attribute marginal distributions between the true statistics in Virginia ACS summary data and samples generated by the categorical maximum entropy model for the attributes *Sex* and *Income*.

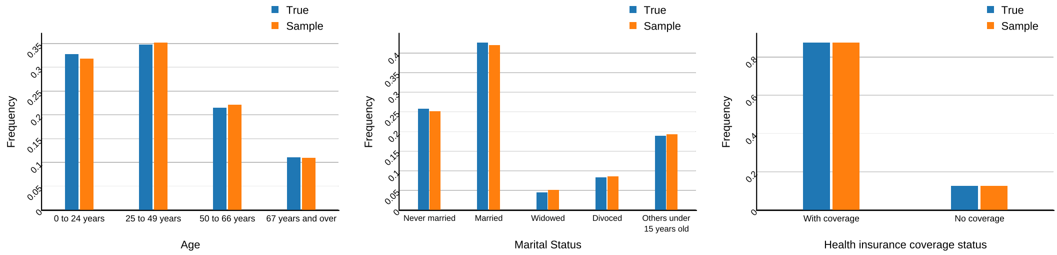

Fig. 6. Comparison of single attribute marginal distributions between the true statistics in Virginia ACS summary data and samples generated by the categorical maximum entropy model for the attributes *Age*, *Marital status*, and *Health insurance coverage status*.

in the Virginia ACS summary data are taken into account. As we can see from the figure, the gain of the log-likelihood of the final heuristic model is quite close to that of the full model, which indicates that the heuristic model discovers and incorporates the majority of the knowledge in the Virginia ACS summary data. Combined with the small value of the approximate KL-divergence measure, these results demonstrate that the proposed categorical maximum entropy model is able to well estimate the categorical data distribution from real data. Table 4 shows the most informative patterns selected by the proposed heuristic.

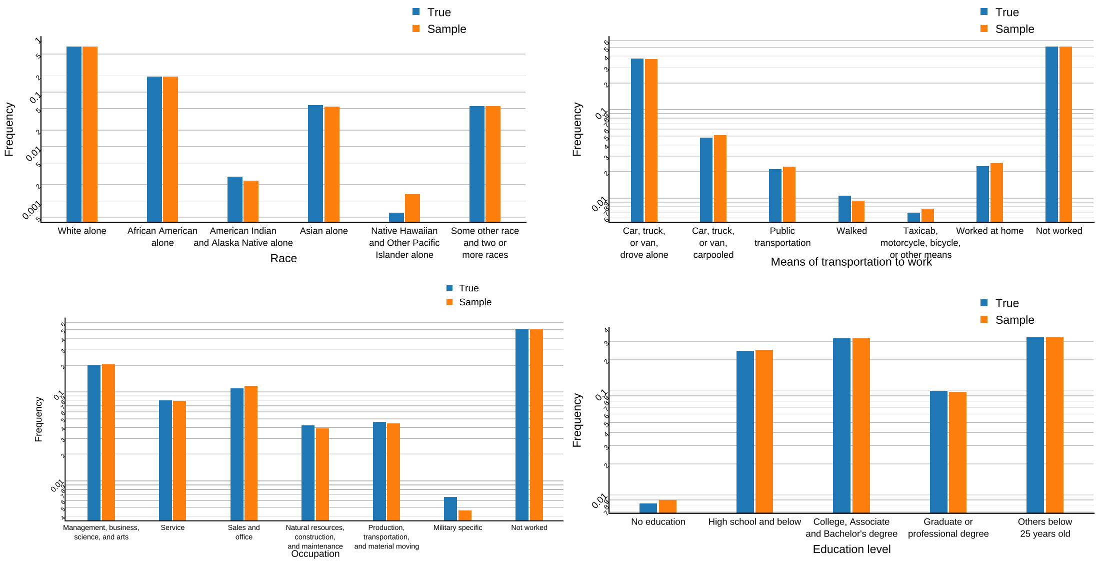

Fig. 7. Comparison of single attribute marginal distributions between the true statistics in Virginia ACS summary data and samples generated by the categorical maximum entropy model for the attributes *Race*, *Means of transportation to work*, *Occupation*, and *Education level*. Y-axis is in log scale.

We also sampled 3, 000 synthetic individuals with the inferred heuristic model for Virginia, and calculated the empirical marginal distributions for all of the individual attributes and all of the multi-attribute categorical patterns that appear in the Virginia ACS summary data. Notice that for attributes *Marital status*, *Means of transportation to work*, *Occupation* and *Education level*, the population considered in the ACS summary data is not the entire population of Virginia state. Thus, we add an additional value for these attributes, e.g. the value *Others under 15 years old* for the attribute *Marital status*, to denote the proportion of the entire population that is not taken into account in the ACS summary data. Figure 5, 6, 7 and 8 show single-attribute and multi-attribute marginal distributions and compare them with the true distributions in the Virginia ACS summary data. We can see that the empirical distributions calculated from the synthetic individuals are very close to those in the Virginia ACS summary data. Such results demonstrate that our categorical maximum entropy model well maintains the statistical characteristics of real world datasets, and is capable of generating synthetic data for real applications.

6.4 Application: Epidemic Simulation

In this section, we apply our proposed categorical maximum entropy model to generate synthetic population for the city of Portland, OR in the United States, and use this model for an epidemiological simulation. We first take a synthetic contact network dataset of Portland [20] that is publicly available. The Portland dataset contains both individual demographic and contact information of residents in the city of Portland. The demographic information in this dataset contains gender, age and household income. We first group the values of age and household income into several ranges and change them into categorical features, similar to our ACS dataset analysis in Section 6.3. Then we compute the statistics, e.g. frequencies, of the single and pairwise demographic features, convert them into categorical patterns, and infer the categorical maximum entropy model over these

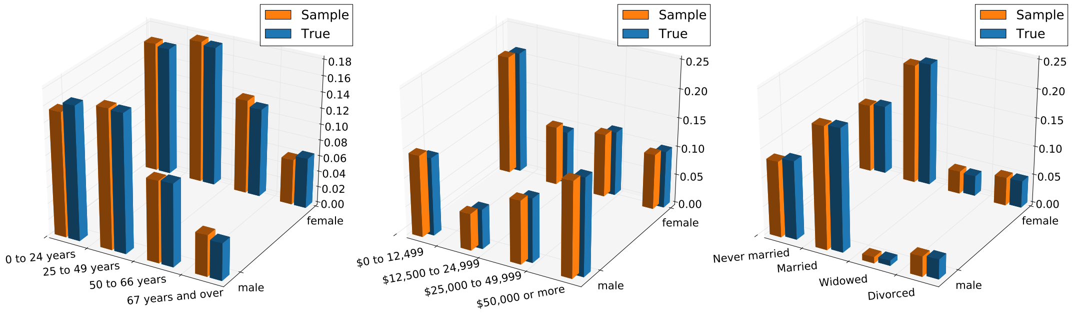

Fig. 8. Comparison of two-attribute marginal distributions between true statistics in the Virginia ACS summary data and samples generated by the categorical maximum entropy model for the categorical patterns $\{$*sex, age*$\}$ (left), $\{$*sex, income*$\}$ (middle), and $\{$*sex, marital status*$\}$ (right). For pattern $\{$*sex, marital status*$\}$, the pattern values whose *marital status* is *Others under 15 years old* is not displayed here since for those individuals, their marital statuses are unavailable.

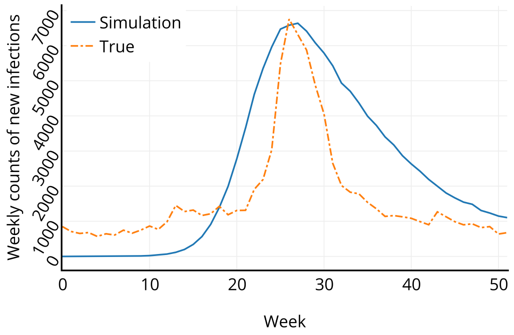

Fig. 9. The simulated weekly flu new infection counts compared to the estimated weekly new infection counts from Google Flu Trends. The simulation results are averaged across 10 independent runs.

patterns. The Portland dataset contains 1, 575, 861 connected individuals, where each individual performs at least one activity with others. To generate our synthetic population, we draw 1, 575, 861 samples from the inferred categorical maximum entropy model.

To construct the contact network for the synthetic population, we first match the generated synthetic individuals to the real ones involved in the contact activities described in the Portland dataset based on their demographical feature values. Then the contact network can be naturally created by connecting the synthetic individuals according to the contact activities they involve in. In this application, we choose to study the flu season in the city of Portland during the period from June 2013 to June 2014. We retrieve the estimated weekly counts of flu new infections for the city of Portland from Google Flu Trends [9], and apply the Susceptible-Infectious (SI) epidemic model over the contact network to fit the curve of weekly flu new infection counts. Figure 9 illustrates the fitted curve using the SI epidemic model. As the figure shows, the simulation results of the SI model over the synthetic population capture the trend and the peak of the weekly flu new infections in the city of Portland. These results demonstrate that the synthetic population generated by the

Table 5. Comparison of our proposed approach (MaxEnt) with the existing synthetic population generation methods.

	IPF	FBS	PeGS	SFG	DMS	BayeNet	**MaxEnt**
Directly capture dependencies beyond two attributes	✓		✓		✓	✓	✓
Directly estimate population joint distribution				✓		✓	✓
Don't require seeding population			✓			✓	✓
Support iterative model inference							✓

categorical maximum entropy model is a useful model of population-level activities in cities. Here, we would like to mention that there are also many other issues, e.g. distance measuring [27, 36, 37], that are relevant when studying disease transmission. However, these topics are beyond the scope of this paper, thus, we will not discuss them in detail here.

7 RELATED WORK

The problem of generating synthetic data that maintain the structures and dependencies in actual data has been studied by researchers from various realms. Iterative proportional fitting (IPF) technique [2] and its variants [18, 38], which generally operate over contingency tables, have been applied to generate synthetic population to study large social contact networks, land use and transportation microsimulation. The NDSSL at Virginia Tech released synthetic datasets of population in the city of Portland [20] generated by a high-performance simulation system *Simfrastructure* which adopts IPF techniques. Such IPF based approaches usually do not directly estimate the joint probability distribution of the data, and sometimes, they require seeding populations as a part of the input. Fitness-based synthesis (FBS) approaches [15] define fitness measures based on control tables to directly generate synthetic populations with seeding data. Sample-free generators (SFG) [3] were proposed to generate synthetic populations using the joint data distribution defined with the data at the most disaggregated level Namazi-Rad et al. [19] applied a dynamic micro-simulation model (DMS) to project dynamics over the synthetic population generated by combinational optimization approaches. Recently, a non-parametric perturbed Gibbs sampler (PeGS) [21] which requires estimating all of the full conditional distributions to represent the joint data distribution was proposed to generate large-scale privacy-safe synthetic health data. Sun and Erath [26] proposed a Bayesian network (BayeNet) based approach to estimate the joint distribution of populations, which is then used to generate synthetic populations. While, our proposed maximum entropy model additionally supports iterative model inference, which makes it easy to update our proposed model with new knowledge about the data. Compared to the proposed categorical maximum entropy model, these existing approaches described above either do not directly capture the dependencies beyond two attributes or do not directly estimate the full joint data distribution. Table 5 compares the proposed maximum entropy approach (MaxEnt) with the related existing synthetic population generation methods in detail.

Maximum entropy models have drawn much attention recently in the pattern mining community, especially in the realm of discovering subjectively interesting patterns. De Bie [7] formalized an information theoretical framework for data mining by applying the maximum entropy principle. In recent research works [8, 23, 32], maximum entropy models were developed to discover and evaluate interesting patterns from binary datasets, and they were also adopted together with the principle of Minimum Description Length to summarize and compress binary datasets [16, 31, 34]. Besides the binary data, maximum entropy modeling was also extended to the multi-relational data and real-valued data [14, 24, 35]. However, all these previous works focus on discovering informative patterns or assessing subjective interestingness of patterns from binary, real-valued, or multi-relational data, and none of them involves categorical data.

In the database community, Gray et al. [10] surveyed several database generation techniques that generate large scale synthetic datasets, and Bruno and Chaudhuri [4] proposed a Data Generation Language (DGL) that allows individual attribute distributions to be specified. A database generation tool that could handle complex inter- and intra-table relationships was proposed by Houkjær et al. [11]. Arasu et al. [1] proposed an efficient, linear programming based algorithm to generate synthetic relational databases that satisfy a given set of declarative constraints. Compared to the proposed approach, these works focus on structured data in relational databases, while our proposed method is generally applicable to categorical data including unstructured categorical data. The maximum entropy principle is also adopted in database query optimization. The sizes of database queries were estimated by modeling complicated database statistics using maximum entropy probability distributions [13, 17]. Ré and Suciu [22] studied the problem of cardinality estimation using the entropy maximization technique with peak approximation. An algorithm called ISOMER was proposed by Srivastava et al. [25] to approximate the true data distribution by applying the maximum entropy principle over database query feedbacks. These works aim to utilize the maximum entropy principle to optimize database queries, while in our method, we are focused on estimating a probabilistic generative model so that synthetic data could be generated.

8 CONCLUSION

In this paper, we have demonstrated a generative probabilistic model for categorical data by employing the maximum entropy principle. By introducing categorical tuple blocks and the corresponding partial order over them, we have presented an efficient model inference algorithm based on the well-known iterative scaling framework. Experiment results on both synthetic data and real US census data show that the proposed model well estimates the underlying categorical data distributions. The application to the problem of epidemic simulation demonstrates that our proposed model can be applied to support research in a variety of applicatin areas.

ACKNOWLEDGMENTS

This work is supported in part by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center (DoI/NBC) contract number D12PC000337, by the National Science Foundation via grants DGE-1545362, IIS-1633363, and by the Army Research Laboratory under grant W911NF-17-1-0021. The US Government is authorized to reproduce and distribute reprints of this work for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, NSF, Army Research Laboratory, or the U.S. Government.

Jilles Vreeken is supported by the Cluster of Excellence “Multimodal Computing and Interaction” within the Excellence Initiative of the German Federal Government.

REFERENCES

[1] Arvind Arasu, Raghav Kaushik, and Jian Li. 2011. Data Generation Using Declarative Constraints. In *SIGMOD ’11*. 685–696.

[2] C.L. Barrett, R.J. Beckman, M. Khan, V. Kumar, M.V. Marathe, P.E. Stretz, T. Dutta, and B. Lewis. 2009. Generation and analysis of large synthetic social contact networks. In *Winter Simulation Conference (WSC)*. 1003–1014.

[3] Johan Barthelemy and Philippe L Toint. 2013. Synthetic population generation without a sample. *Transportation Science* 47, 2 (2013), 266–279.

[4] Nicolas Bruno and Surajit Chaudhuri. 2005. Flexible Database Generators. In *VLDB ’05*. VLDB Endowment, 1097–1107.

[5] Imre Csiszár. 1975. I-Divergence Geometry of Probability Distributions and Minimization Problems. *Annals of Probability* 3, 1 (1975), 146–158.

[6] J. N. Darroch and D. Ratcliff. 1972. Generalized Iterative Scaling for Log-Linear Models. *The Annals of Mathematical Statistics* 43, 5 (1972), 1470–1480.

[7] Tijl De Bie. 2011. An information theoretic framework for data mining. In *proceeding of KDD ’11*. ACM, 564–572.

[8] Tijl De Bie. 2011. Maximum entropy models and subjective interestingness: an application to tiles in binary databases. *Data Min. Knowl. Discov.* 23, 3 (Nov. 2011), 407–446.

[9] Google Inc. 2014. Data Source: Google Flu Trends. (2014). http://www.google.org/flutrends.

[10] Jim Gray, Prakash Sundaresan, Susanne Englert, Ken Baclawski, and Peter J. Weinberger. 1994. Quickly Generating Billion-record Synthetic Databases. In *SIGMOD ’94*. ACM, 243–252.

[11] Kenneth Houkjær, Kristian Torp, and Rico Wind. 2006. Simple and Realistic Data Generation. In *VLDB ’06*. 1243–1246.

[12] W. Hu, L. Yan, and H. Wang. 2014. Traffic jams prediction method based on two-dimension cellular automata model. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. 2023–2028.

[13] Raghav Kaushik, Christopher Ré, and Dan Suciu. 2009. General Database Statistics Using Entropy Maximization. In *DBPL ’09*. Springer-Verlag, 84–99.

[14] Kleanthis-Nikolaos Kontonasios, Jilles Vreeken, and Tijl De Bie. 2013. Maximum Entropy Models for Iteratively Identifying Subjectively Interesting Structure in Real-Valued Data. In *ECMLPKDD ’13*. Springer, 256–271.

[15] Lu Ma and Sivaramakrishnan Srinivasan. 2015. Synthetic Population Generation with Multilevel Controls: A Fitness-Based Synthesis Approach and Validations. *Computer-Aided Civil and Infrastructure Engineering* 30, 2 (2015), 135–150.

[16] Michael Mampaey, Nikolaj Tatti, and Jilles Vreeken. 2011. Tell me what I need to know: succinctly summarizing data with itemsets. In *KDD ’11*. ACM, 573–581.

[17] V. Markl, N. Megiddo, M. Kutsch, T. M. Tran, P. Haas, and U. Srivastava. 2005. Consistently Estimating the Selectivity of Conjuncts of Predicates. In *VLDB ’05*. 373–384.

[18] Kirill Mueller and Kay W Axhausen. 2011. Hierarchical IPF: Generating a synthetic population for Switzerland. In *ERSA conference papers*. European Regional Science Association.

[19] Mohammad-Reza Namazi-Rad, Payam Mokhtarian, and Pascal Perez. 2014. Generating a Dynamic Synthetic Population–Using an Age-Structured Two-Sex Model for Household Dynamics. *PloS one* 9, 4 (2014).

[20] Network Dynamics and Simulation Science Laboratory. 2015. *Synthetic Data Products for Societal Infrastructures and Proto-Populations: Data Set 2.0.* Technical Report. Virginia Polytechnic Institute and State University. NDSSL-TR-07-003.

[21] Y. Park, J. Ghosh, and M. Shankar. 2013. Perturbed Gibbs Samplers for Generating Large-Scale Privacy-Safe Synthetic Health Data. In *ICHI’13*. 493–498.

[22] Christopher Ré and Dan Suciu. 2010. Understanding Cardinality Estimation Using Entropy Maximization. In *PODS ’10*. ACM, 53–64.

[23] Koen Smets and Jilles Vreeken. 2012. Slim: Directly Mining Descriptive Patterns. In *SDM ’12*. SIAM, 236–247.

[24] Eirini Spyropoulou, Tijl De Bie, and Mario Boley. 2014. Interesting pattern mining in multi-relational data. *Data Min. Knowl. Discov.* 28, 3 (2014), 808–849.

[25] U. Srivastava, P. J. Haas, V. Markl, M. Kutsch, and T. M. Tran. 2006. ISOMER: Consistent Histogram Construction Using Query Feedback. In *ICDE ’06*. IEEE Computer Society.

[26] Lijun Sun and Alexander Erath. 2015. A Bayesian network approach for population synthesis. *Transportation Research Part C: Emerging Technologies* 61 (2015), 49 – 62.

[27] M. Tan, B. Wang, Z. Wu, J. Wang, and G. Pan. 2016. Weakly Supervised Metric Learning for Traffic Sign Recognition in a LIDAR-Equipped Vehicle. *IEEE Transactions on Intelligent Transportation Systems* 17, 5 (May 2016), 1415–1427.

[28] D. Tao, X. Li, X. Wu, and S. J. Maybank. 2007. General Tensor Discriminant Analysis and Gabor Features for Gait Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29, 10 (Oct 2007), 1700–1715.

[29] D. Tao, X. Li, X. Wu, and S. J. Maybank. 2009. Geometric Mean for Subspace Selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31, 2 (Feb 2009), 260–274.

[30] Nikolaj Tatti. 2006. Computational complexity of queries based on itemsets. *Inform. Process. Lett.* 98, 5 (2006), 183–187.

[31] Nikolaj Tatti and Jilles Vreeken. 2008. Finding Good Itemsets by Packing Data. In *ICDM '08*. IEEE, 588–597.

[32] Nikolaj Tatti and Jilles Vreeken. 2011. Comparing Apples and Oranges: Measuring Differences between Data Mining Results. In *proceeding of ECMLPKDD '11*. Springer, 398–413.

[33] United States Census Bureau. 2012. American Community Survey. (2012). http://www.census.gov/acs/www/

[34] Jilles Vreeken, Matthijs van Leeuwen, and Arno Siebes. 2011. Krimp: Mining Itemsets that Compress. *Data Min. Knowl. Discov.* 23, 1 (2011), 169–214.

[35] Hao Wu, Jilles Vreeken, Nikolaj Tatti, and Naren Ramakrishnan. 2014. Uncovering the Plot: Detecting Surprising Coalitions of Entities in Multi-relational Schemas. *Data Min. Knowl. Discov.* 28, 5-6 (Sept. 2014), 1398–1428.

[36] J. Yu, Y. Rui, Y. Y. Tang, and D. Tao. 2014. High-Order Distance-Based Multiview Stochastic Learning in Image Classification. *IEEE Transactions on Cybernetics* 44, 12 (Dec 2014), 2431–2442.

[37] J. Yu, X. Yang, F. Gao, and D. Tao. 2017. Deep Multimodal Distance Metric Learning Using Click Constraints for Image Ranking. *IEEE Transactions on Cybernetics* PP, 99 (2017), 1–11.

[38] Yi Zhu and Joseph Ferreira. 2014. Synthetic Population Generation at Disaggregated Spatial Scales for Land Use and Transportation Microsimulation. *Journal of the Transportation Research Board* 2429, 1 (2014), 168–177.

Received February 2017; revised October 2017; accepted Janaury 2018