Linear Space Direct Pattern Sampling using Coupling From The Past

Mario Boley
Fraunhofer IAIS and
University of Bonn
mario.boley@iais.fhg.de

Sandy Moens
University of Antwerp
sandy.moens@ua.ac.be

Thomas Gärtner
Fraunhofer IAIS and
University of Bonn
thomas.gaertner@iais.fhg.de

ABSTRACT

This paper shows how coupling from the past (CFTP) can be used to avoid time and memory bottlenecks in direct local pattern sampling procedures. Such procedures draw controlled amounts of suitably biased samples directly from the pattern space of a given dataset in polynomial time. Previous direct pattern sampling methods can produce patterns in rapid succession after some initial preprocessing phase. This preprocessing phase, however, turns out to be prohibitive in terms of time and memory for many datasets. We show how CFTP can be used to avoid any super-linear preprocessing and memory requirements. This allows to simulate more complex distributions, which previously were intractable. We show for a large number of public real-world datasets that these new algorithms are fast to execute and their pattern collections outperform previous approaches both in unsupervised as well as supervised contexts.

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

Keywords: Local patterns, Sampling, CFTP, Frequent sets

1. INTRODUCTION

This paper presents a unified sampling algorithm for efficiently drawing local patterns [11] directly from the pattern space of a given input dataset. Traditionally, local pattern discovery problems are often addressed by branch-and-bound search algorithms (e.g., for association discovery [16] or subgroup discovery [10]). These methods allow to exhaustively inspect a high-frequency fraction of the pattern space that is limited by the available computation time and sometimes memory (e.g., for best-first-search priority queues or candidate pruning). This can be problematic for two reasons: 1) Although it is reasonable to prune patterns of very low frequency, for the remaining patterns, frequency is not generally a good ranking criterion. 2) There is a high level of redundancy among the contained patterns, i.e., all local phenomena are approximately described by many alternative descriptions. While both factors seem to make the discovery task very hard, in particular the redundancy allows *random pattern collections* to perform robustly and well in many applications. For instance, suppose we are interested in combinations of low-frequency singletons (items) that have a relatively high conjunctive frequency. Then, instead of exhaustively listing a huge part of the pattern space, it can be much more efficient to just draw a small number of patterns according to a distribution $\mathcal{F}$ with, e.g.,

$$\mathcal{F}(F) = \mathsf{frq}^2(F) \prod_{e \in F} (1 - \mathsf{frq}(\{e\})) \ . \tag{1}$$

Direct pattern sampling refers to the idea of drawing such random collections directly from the pattern space—in polynomial time without physically materializing auxiliary parts of the space (see Figure 1(a)).

1.1 Two-step Direct Pattern Sampling

Many instances of direct pattern sampling can be realized by a simple two-step random procedure, which is a generalization of several previous algorithms for specific distributions [4]. The procedure is applicable if the desired distribution can be expressed as a product of singleton prior weights and a small number of factors based on variants of frequency. For the example given in Eq. (1) this holds because it consists of two frq-factors and multiplicative singletons weights $b(e) = 1 - \mathsf{frq}(\{e\})$. Many other variants are possible, including products of area, lift, disjunctive frequency, various discriminativity measures, and additive singleton weights (see Section 3). The two-step approach can be outlined as follows (see also Figure 1(b)):

Step 1 draw from the input data a tuple of data records of dimensionality equal to the number of frequency factors (with probability proportional to the total prior weight of all patterns it contains); and

Step 2 draw a pattern contained in that tuple according to the prior weights.

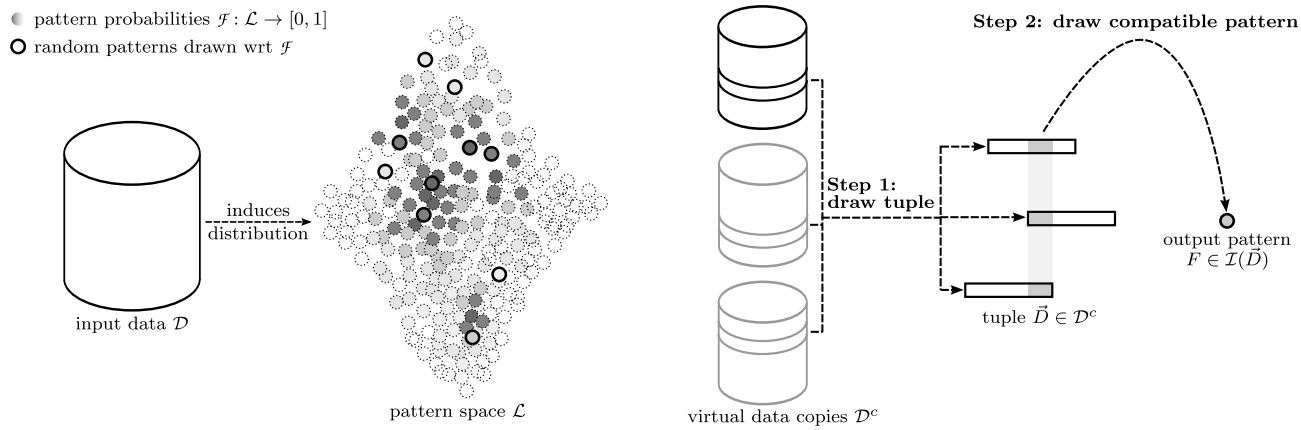
For the example of Eq. (1), Step 1 corresponds to drawing a pair of data records (D_1, D_2) according to the total multiplicative pattern weight

$$w(D_1, D_2) = \sum_{F \subseteq (D_1 \cap D_2)} \prod_{e \in F} (1 - \mathsf{frq}(\{e\})) \ .$$

Step 2 corresponds to drawing a pattern F supported by both, D_1 and D_2, according to $\prod_{e \in F}(1 - \mathsf{frq}(\{e\}))$.

In contrast to the cited branch-and-bound algorithms or earlier approaches to pattern sampling based on Markov

KDD'12, August 12–16, 2012, Beijing, China.



(a) direct pattern sampling

(b) two-step sampling framework

Figure 1: Direct pattern sampling (a) means to directly materialize a set of random patterns from input data wrt to a distribution that favors useful patterns. Two-step sampling (b) is an approach to direct pattern sampling for distributions that are a product of c (negative) frequency factors: in Step 1 a tuple of c data records is drawn and in Step 2 a pattern is generated that is (not) supported by all elements of that tuple.

chain Monte Carlo [1, 3, 5], this two-step sampling framework can be implemented as a proper polynomial time algorithm. However, the direct (enumerative) implementation of Step 1 requires the availability of the weights w of all tuples of data records. Generally, for c frequency factors this corresponds to a data structure of size $|\mathcal{D}|^c$, which has to be constructed in a pre-processing phase. Depending on the available hardware, this renders distributions with two factors already infeasible for datasets with between 10^4 and 10^5 data records. Distributions with three factors are only feasible for very small datasets of at most a few 100 records.

1.2 Outline and Contributions

In this paper we substantially push the limits of the distributions one can practically simulate by replacing the enumerative implementation of Step 1 by an indirect approach based on coupling from the past[1] (CFTP [13, 18]; Sections 2 and 4). This allows to omit any super-linear preprocessing and memory requirements. Without this bottleneck a much wider range of datasets becomes accessible for simulating distributions with two factors and even with three and four factors. This is demonstrated empirically with 47 public real-world datasets. For many distributions we also show a theoretical polynomial upper bound on the expected running time. To fully utilize the power of this new algorithm, we lift the two-step sampling idea to a general framework and instantiate it with 17 different distributions based on pattern frequency, area, rare singletons, and label discriminativity (Section 3). We evaluate these distributions in both, an unsupervised context based on the compression-based pattern selector Krimp [19] and a supervised context based on a selection measure used in pattern-based classification (e.g., [6]). Our experiments show that the distributions that are enabled by the techniques of this paper, are indeed valuable for discovering patterns of greater utility: they outperform previously available distributions as well as top-k frequent closed set mining by a wide margin.

[1]See http://www-kd.iai.uni-bonn.de/index.php?page=software.

2. PRELIMINARIES

Before we present the technical contributions of the paper, we give a comprehensive summary of local pattern discovery and CFTP. Also we fix some basic notational conventions. We denote random variables by boldface letters, e.g., $\boldsymbol{r}$, and distributions by stylized letters, e.g., $\mathit{d}: \Omega \to [0,1]$. The **uniform distribution** on Ω is denoted $u[\Omega]$. For the most part we are interested in finite domains Ω. In this case, we are identifying distributions with row vectors, and, as a convention, for a set of positive weights $w: \Omega \to \mathbb{R}_+$ we denote by w the distribution on Ω resulting from normalizing the weights w, i.e., $w(x) = w(x)/\sum_{x' \in \Omega} w(x')$. For two distributions d, d' on finite Ω we denote by $\|\mathit{d}, \mathit{d}'\|_{\mathrm{tv}} = 1/2 \sum_{x \in \Omega} |\mathit{d}(x) - \mathit{d}'(x)|$ the **total variation distance** between d and d'. When extending real-valued functions—in particular weight and probability functions—to sets, we interpret them additively, i.e., $w(\Omega') = \sum_{x \in \Omega'} w(x)$ for $\Omega' \subseteq \Omega$. For the **image set** of arbitrary functions f with domain X we write $\mathrm{img}(f) = \{f(x) \colon x \in X\}$. The **power set** of a set X is denoted $\mathcal{P}(X)$.

2.1 Local Pattern Discovery

A binary **dataset** $\mathcal{D}$ over some finite set E of **singletons** (items) is a bag (multiset) of sets, called **data records**, $D_1, \ldots, D_m$ each of which is a subset of E. The **size** of $\mathcal{D}$, denoted by $\|\mathcal{D}\|$, is defined as the sum of all its data record sizes $\sum_{D \in \mathcal{D}} = |D|$. For a given dataset $\mathcal{D}$ over E, the **pattern space** (or pattern *language*) $\mathcal{L}(\mathcal{D})$ considered in this paper is the family of non-empty non-singleton subsets of E, i.e., $\mathcal{L}(\mathcal{D}) = \{F \subseteq E \colon |F| \geq 2\}$. The elements of $\mathcal{L}$ are interpreted conjunctively. That is, the *local* data portion described by a set $F \subseteq E$, called the **support (set)** of F in $\mathcal{D}$ and denoted $\mathcal{D}[F]$, is defined as the multiset of all data records from $\mathcal{D}$ that contain *all* elements of F, i.e., $\mathcal{D}[F] = \{D \in \mathcal{D} \colon D \supseteq F\}$.

We recap some basic measures of pattern utility that we use in this paper. The most fundamental measures for set patterns are the **support (count)**, i.e., the size of its support set $\mathsf{supp}(\mathcal{D}, F) = |\mathcal{D}[F]|$, and the **frequency**, i.e., the

relative size of its support with respect to the total number of data records $\mathsf{frq}(\mathcal{D}, F) = |\mathcal{D}[F]| \,/\, |\mathcal{D}|$. Correspondingly, we define the **negative support** and the **negative frequency** as $\overline{\mathsf{supp}}(\mathcal{D}, F) = \left|\overline{\mathcal{D}}[F]\right|$ and $\overline{\mathsf{frq}}(\mathcal{D}, F) = \left|\overline{\mathcal{D}}[F]\right| / |\mathcal{D}|$, respectively, where $\overline{\mathcal{D}}[F] = \mathcal{D} \backslash \mathcal{D}[F]$ denotes the complement of the support set of F. A further measure considered here is the **area function** [9] $\mathsf{area}(\mathcal{D}, F) = |F|\,|\mathcal{D}[F]|$. In some application contexts, e.g., subgroup discovery or emerging pattern mining [7], each data record $D \in \mathcal{D}$ has an associated **class label** $l(D) \in \{\oplus, \ominus\}$. We denote by $\mathcal{D}_l \in \{\oplus, \ominus\}$ the data portion labeled l, i.e., $\mathcal{D}_l = \{D \in \mathcal{D} \colon l(D) = l\}$. For such datasets we consider as a representative utility measure the **information gain**, which is defined by

$$\mathsf{ig}(\mathcal{D}, F) = H(\mathcal{D}) - \mathsf{frq}(\mathcal{D}, F) H(\mathcal{D}[F]) - \overline{\mathsf{frq}}(\mathcal{D}, F) H(\overline{\mathcal{D}}[F])$$

where for a data portion $\mathcal{D}' \subseteq \mathcal{D}$

$$H(\mathcal{D}') = -\sum_{l \in \{\oplus, \ominus\}} \left|\mathcal{D}'_l\right| / \left|\mathcal{D}'\right| \log \left|\mathcal{D}'_l\right| / \left|\mathcal{D}'\right|$$

is the entropy of the label distribution associated with $\mathcal{D}'$. Note that all concepts discussed here can be easily extended to the case of more than two classes.

2.2 Coupling From The Past

Coupling from the past is a technique to acquire a sample from a distribution d on some domain Ω without explicitly constructing it. It is based on an indirect simulation of a Markov chain on Ω with a state distribution that converges to d. Within the two-step sampling framework of this paper, it is used to efficiently implement Step 1.

A (time-homogeneous) Markov chain on finite state space Ω is a discrete time random process that can be specified by a stochastic **state transition matrix** $P \in [0,1]^{\Omega \times \Omega}$. The chain is called **ergodic** if for all $x, y \in \Omega$ there is a number $t_0 \in \mathbb{N}$ of steps such that for all $t > t_0$ it holds that $P^t_{x,y} > 0$ ($P^t_{x,y}$ is equal to the probability of going from x to y in t steps). An ergodic Markov chain has a **stationary distribution**, i.e., a distribution $\mathit{d} \colon \Omega \to [0,1]$ with $\mathit{d}P = \mathit{d}$, that it converges to, i.e., for all distributions d' $\lim_{t\to\infty} \|\mathit{d}' P^t, \mathit{d}\|_{\mathrm{tv}} = 0$. Given a desired target distribution d on Ω, a Markov chain with stationary distribution d can be constructed by using as transition procedure the **Metropolis-Hastings algorithm** [12]: in a current state x propose a successor state y with some proposal distribution $q_x \colon \Omega \to [0,1]$ and then accept the proposal y as the new current state if $\boldsymbol{u} \leq (\mathit{d}(y) q_y(x)) / (\mathit{d}(x) q_x(y))$ with uniform $\boldsymbol{u} \sim u[[0,1)]$ and otherwise keep x as current. The resulting transition probabilities are

$$P_{x,y} = q_x(y) \min\left(\frac{\mathit{d}(y) q_y(x)}{\mathit{d}(x) q_x(y)}, 1\right) \tag{2}$$

if $q_x(y) > 0$ and 0 otherwise. If the Markov chain described by P is ergodic the stationary distribution is the desired d.

In order to acquire a sample of the stationary distribution one needs an efficient single step simulation algorithm that computes a **random transition map** $\phi(\cdot, \boldsymbol{r}) \colon \Omega \to \Omega$ with a suitable random variable $\boldsymbol{r}$ such that

$$\forall x, y \in \Omega,\ \mathbb{P}[\phi(x, \boldsymbol{r}) = y] = P_{x,y} \quad . \tag{3}$$

One way of sampling is then to run a **forward simulation** for t time steps from some starting state x_0, i.e., to compute $(\phi(\cdot, \boldsymbol{r}_t) \circ \cdots \circ \phi(\cdot, \boldsymbol{r}_1))(x_0)$ where the $\boldsymbol{r}_i$ are i.i.d. copies of $\boldsymbol{r}$. This is what is commonly referred to as Markov chain Monte Carlo method. Disadvantages of this approach are that it yields only approximate samples of d and that it is hard to come up with good a priori bounds for the required number of steps t. **Coupling from the past** (CFTP) is an alternative method that avoids these drawbacks. It is based on **backward simulations** defined by $\Phi_i = \Phi_{i-1} \circ \phi(\cdot, \boldsymbol{r}_i)$ for $i > 0$ and $\Phi_0(x) = x$. The crucial insight is: if for some $t > 0$ it holds that Φ_t is a constant function, i.e.,

$$\mathrm{img}(\Phi_t) = \{\boldsymbol{x}\} \tag{4}$$

then $\boldsymbol{x}$ must be an observation of the stationary distribution ($\mathbb{P}[\boldsymbol{x} = x] = \mathit{d}(x)$). This event is referred to as **coalescence**. Intuitively we can consider the backward simulation to be running already since infinitely far in the past, hence the current state is distributed according to d, but for determining the current state no information beyond the **time horizon** t is required. This is because Eq. (4) corresponds to the event that all possible realizations of the Markov chain agree in their current state (using all possible starting states but fixing the source of randomness for state transition). The CFTP protocol checks Eq. (4) for Φ_t in exponentially increasing **epochs**, i.e., $t = 2^i$ for $i = 1, 2, \ldots$. The challenge is how to efficiently compute $\mathrm{img}(\Phi_t)$. Naively keeping track of all realizations would diminish any advantage over enumerative sampling (explicitly constructing Ω). Sec. 4 shows how this can be achieved for our specific sampling task.

3. TWO-STEP DIRECT SAMPLING

In this section we present a general form of the two-step sampling framework and show how it can be instantiated to simulate various distributions relevant to local pattern discovery.

3.1 Distributions

Given an input dataset $\mathcal{D}$ over E, the framework can be used to sample according to distributions $\mathcal{F} \colon \mathcal{L}(\mathcal{D}) \to [0,1]$ that can be expressed in the following product form:

$$\mathcal{F}(F) = b_\star(F) \prod_{i=1}^{c} q_i(\mathcal{D}_i, F) / Z \tag{5}$$

where Z is a normalizing constant, $q_i \in \{\mathsf{supp}, \overline{\mathsf{supp}}\}$ are support resp. negative support measures for some specific portions of the input data $\mathcal{D}_i \subseteq \mathcal{D}$, and $b_\star \colon \mathcal{P}(E) \to \mathbb{R}_+$ is a weight function based on positive singleton prior weights $b \colon E \to \mathbb{R}_+$ that are interpreted either multiplicative or additive, i.e., $b_\star(F) = \bigstar_{e \in F} b(e)$ with $\bigstar \in \{\Pi, \Sigma\}$.

Out of the many possible distributions that can be represented with Eq. (5), in this paper, we consider 17 representative examples with between one and four factors. The distributions can be categorized into four groups, each of which has a base-case with one or two factors, respectively. Distributions with a greater number of factors result then from multiplying the base-case with more frequency-factors. The groups are:

Frequency with base-case $\mathcal{F}_{\mathsf{frq}}(F) = \mathsf{frq}(\mathcal{D}, F)/Z$ by setting $q_1 = \mathsf{supp}$, $\mathcal{D}_1 = \mathcal{D}$, $b(e) = 1$, and $\star = \Pi$ (creating uniform priors $b_\star(\cdot) = 1$); and distributions $\mathcal{F}_{\mathsf{frq}^2}$, $\mathcal{F}_{\mathsf{frq}^3}$, $\mathcal{F}_{\mathsf{frq}^4}$ by setting factors accordingly.

Area with base-case $\mathcal{F}_{\mathsf{area}}(F) = \mathsf{area}(\mathcal{D}, F)/Z$ by setting $q_1 = \mathsf{supp}$, $\mathcal{D}_1 = \mathcal{D}$, $b(e) = 1$, and $\star = \Sigma$; and distributions $\mathcal{F}_{\mathsf{ar}\cdot\mathsf{fq}}$, $\mathcal{F}_{\mathsf{ar}\cdot\mathsf{fq}^2}$, $\mathcal{F}_{\mathsf{ar}\cdot\mathsf{fq}^3}$.

Algorithm 1 Two-Step Sampling Framework

Require: data portions $\mathcal{D}_1, \dots, \mathcal{D}_c$ of input data $\mathcal{D}$ over E, measures $q_1, \dots, q_c \in \{\mathsf{supp}, \overline{\mathsf{supp}}\}$, singleton weights $b\colon E \to \mathbb{R}_+$ and $\star \in \{\Pi, \Sigma\}$
Returns: random pattern $\boldsymbol{F} \sim \mathcal{F}$ as in Eq. (5)

1. **draw** tuple $\vec{\boldsymbol{D}} \sim w\colon \mathfrak{D} \to [0,1]$ with weights
$$w(\vec{D}) = b_\star(\mathcal{I}(\vec{D})) - b_\star(\overline{\mathcal{L}}(\vec{D}))$$
2. **draw** result pattern $\boldsymbol{F} \sim b'_\star\colon \mathcal{L} \to [0,1]$ with
$$b'_\star(F) = \begin{cases} b_\star(F) & \text{, if } F \in \mathcal{I}(\vec{\boldsymbol{D}}) \text{ and } |F| \geq 2 \\ 0 & \text{, otherwise} \end{cases}$$

Rare containing the example from the introduction with base-case $\mathcal{F}_{\mathsf{rare}}(F) = \mathsf{frq}(\mathcal{D}, F) \prod_{e \in F} \overline{\mathsf{frq}}(\mathcal{D}, \{e\})/Z$ by setting $q_1 = \mathsf{supp}$, $\mathcal{D}_1 = \mathcal{D}$, $b(e) = \overline{\mathsf{frq}}(\mathcal{D}, \{e\})$ and $\star = \Pi$; and distributions $\mathcal{F}_{\mathsf{rr{\cdot}fq}}$, $\mathcal{F}_{\mathsf{rr{\cdot}fq}^2}$, $\mathcal{F}_{\mathsf{rr{\cdot}fq}^3}$.

Discriminativity containing distributions biased towards patterns with a high label discriminativity with base-case $\mathcal{F}_{\mathsf{dscr}}(F) = \mathsf{frq}(\mathcal{D}_\oplus, F)\overline{\mathsf{frq}}(\mathcal{D}_\ominus, F)$ by $q_1 = \mathsf{supp}$, $q_2 = \overline{\mathsf{supp}}$, $\mathcal{D}_1 = \mathcal{D}_\oplus$, $\mathcal{D}_2 = \mathcal{D}_\ominus$; and distributions $\mathcal{F}_{\mathsf{dr{\cdot}fq}}$ and $\mathcal{F}_{\mathsf{dr{\cdot}fq}^2}$ as well as distributions resulting from multiplying with frequency within the positively labeled data, i.e., $\mathcal{F}_{\mathsf{dr{\cdot}fq}_+}$ and $\mathcal{F}_{\mathsf{dr{\cdot}fq}^2_+}$.

Note that, while enumerating patterns in descending order of utility is NP-hard for the underlying measures of the area and the discriminativity group, using them as a bias for random pattern collections is possible in polynomial time using the sampling framework presented below.

3.2 Algorithm

The underlying idea of the algorithm follows a simple intuition: a pattern (not) supported by a random data record is likely to be (not) supported by many data records altogether. This principle can be further extended by considering patterns that are simultaneously (not) supported by independent data records drawn from the respective data portions. Algorithmically this can be realized in two steps: Step 1 is to draw a tuple of data records from the Cartesian product of the c data portions and then Step 2 is to draw a pattern compatible with the complete tuple according to its weight $b_\star$. Note that in Step 1 the tuple must be sampled according to the total weight of patterns compatible with it.

In order to precisely formalize this approach, let us introduce some further notation. We denote by $\mathfrak{D} = \times_{i=1}^{c} \mathcal{D}_i$ the **data product**, i.e., the set of all tuples that can be drawn in Step 1. Moreover, let $P \subseteq \{1, \dots, c\}$ be the set of **positive indices**, i.e., $i \in \{1, \dots, c\}$ with $q_i = \mathsf{supp}$ and correspondingly N the set of **negative indices**, i.e., $j \in \{1, \dots, c\}$ with $q_j = \overline{\mathsf{supp}}$. Then a pattern $F \in \mathcal{L}$ is **compatible** with a tuple $\vec{D} \in \mathfrak{D}$ if it satisfies all support requirements, i.e., $F \subseteq \vec{D}(i)$ for all $i \in P$ and $F \not\subseteq \vec{D}(j)$ for all $j \in N$. We refer to the family of all patterns compatible with a tuple $\vec{D}$ as the patterns **induced** by $\vec{D}$ and denote it $\mathcal{I}(\vec{D})$

$$= \{F \subseteq E\colon (\forall i \in P, F \subseteq \vec{D}(i)) \wedge (\forall j \in N, F \not\subseteq \vec{D}(j))\} \ .$$

Finally let us denote the **induced non-pattern sets** by a $\vec{D} \in \mathfrak{D}$ as $\overline{\mathcal{L}}(\vec{D}) = \{F \in \mathcal{I}(\vec{D})\colon |F| \leq 1\}$. With this we can express the weights $w\colon \mathfrak{D} \to R_+$ according to which the tuples must be sampled in Step 1 as

$$w(\vec{D}) = b_\star(\mathcal{I}(\vec{D})) - b_\star(\overline{\mathcal{L}}(\vec{D})) \ .$$

The complete sampling procedure is summarized in the formal pseudo-code given in Alg. 1. Before we turn to the efficiency of the algorithm we first note its correctness.

Proposition 1. *Given data portions $\mathcal{D}_1, \dots, \mathcal{D}_c \subseteq \mathcal{D}$, $q_1, \dots, q_c \in \{\mathsf{supp}, \overline{\mathsf{supp}}\}$, singleton weights $b\colon E \to \mathbb{R}_+$ and $\star \in \{\Pi, \Sigma\}$, Algorithm 1 returns a random pattern $\boldsymbol{F} \in \mathcal{L}(\mathcal{D})$ according to $\mathcal{F}$ as specified in Equation (5).*

Proof. We show that every pattern $F \in \mathcal{L}$ has the correct marginal probability with respect to the joint distribution of all pairs of inducing tuples and patterns drawn in line 1 and 2 of the algorithm, respectively. Let us denote by $\mathcal{I}^-(F) = \{\vec{D} \in \mathfrak{D}\colon F \in \mathcal{I}(\vec{D})\}$ the inverse induces-relation. Then we have

$$\begin{aligned} \mathbb{P}[\boldsymbol{F} = F] &= \sum_{\vec{D} \in \mathfrak{D}} \mathbb{P}[\vec{\boldsymbol{D}} = \vec{D}, \boldsymbol{F} = F] \\ &= \sum_{\vec{D} \in \mathcal{I}^-(F)} \frac{w(\vec{D})}{Z} \frac{b_\star(F)}{w(\vec{D})} \\ &= \left|\{\vec{D} \in \mathfrak{D}\colon F \in \mathcal{I}(\vec{D})\}\right| b_\star(F)/Z \\ &= b_\star(F) \prod_{i=1}^{c} q_i(\mathcal{D}_i, F)/Z = \mathcal{F}(F) \end{aligned}$$

with $Z = \sum_{\vec{D} \in \mathfrak{D}} b'_\star(\mathcal{I}(\vec{D}))$ being the sum of all prior weights of patterns among all inducing tuples. □

3.3 Implementation

We now describe how Algorithm 1 can be implemented efficiently. Step 1 requires to sample a tuple according to its weight. Since there are at most $|\mathcal{D}|^c$ inducing tuples, for some datasets this can be done enumeratively, i.e., by constructing a list of all weights $w(\vec{D}_1), \dots, w(\vec{D}_{m^c})$, draw a random real $\boldsymbol{u} \sim u[[0,1)]$ and then return the unique $\vec{D}_i \in \mathfrak{D}$ with $\sum_{i=1}^{k-1} w(\vec{D}_i) \leq \boldsymbol{u} < \sum_{i=1}^{k} w(\vec{D}_i)$. This is what we refer to as the **enumerative baseline** for Step 1 [4]. In Section 4 we present a much more effective realization based on CFTP. However, both approaches require a method to efficiently compute the weights w. By the inclusion-exclusion formula

$$\begin{aligned} &b_\star(\mathcal{I}(\vec{D})) = \\ &b_\star(\mathcal{P}(\bigcap_{i \in P} \vec{D}(i))) + \sum_{\emptyset \neq I \subseteq N} (-1)^{|I|} b_\star(\mathcal{P}(\bigcap_{j \in I \cup P} \vec{D}(j))) \end{aligned} \quad (6)$$

the total weight of the family of patterns induced by some tuple $\vec{D}$ can be reduced to computing the total weights of power sets $\mathcal{P}(S)$ with $S \subseteq E$. This is independent of the choices of $\star$. Computing the total weight of all subsets of a set $S \subseteq E$ differs based on whether we have multiplicative or additive weights. For multiplicative weights we can use

$$b_\Pi(\mathcal{P}(S)) = \prod_{s \in S} (1 + b(e)) \quad (7)$$

and for additive weights the formula is

$$b_\Sigma(\mathcal{P}(S)) = b_\Sigma(S) 2^{|S|-1} \ . \quad (8)$$

Both can be shown straightforwardly by induction on $|S|$. Note that although the evaluation of Equation 6 requires

Algorithm 2 Sequential Step 2

Require: tuple $\vec{D} \in \mathfrak{D}$,
Returns: random pattern $\boldsymbol{F} \sim \mathscr{b}'_\star$ as in Step 2 of Alg. 1

1. $X \leftarrow \{\}, Y \leftarrow \{\}$
2. **for** $e \in E$ **do**
3. with prob. $\mathbb{P}_{\boldsymbol{F}\sim\mathscr{b}'_\star}[e \in \boldsymbol{F} \mid X \subseteq \boldsymbol{F}, Y \cap \boldsymbol{F} = \emptyset]$ **set** $X \leftarrow X \cup \{e\}$; otherwise **set** $Y \leftarrow Y \cup \{e\}$
4. **return** X

the evaluation of $2^{|N|}$ terms, this does not pose a problem for the small values of $c \geq |N|$ we are usually interested in.

For the implementation of Step 2 we need to sample a pattern according to $\mathscr{b}'_\star$, i.e., a pattern F according to $b_\star(F)$ that is compatible with $\vec{\boldsymbol{D}}$ drawn in Step 1. As a single tuple can induce $2^{|E|}$ patterns, enumerative sampling is not an option for this step even for small datasets. Instead one can use the following sequential selection process on the singletons $e \in E$ that keeps track of the set X of already selected and the set Y of already considered but discarded singletons (see Alg. 2): As long as one can find a singleton $e \notin (X \cup Y)$, add it to the current solution X with probability

$$\mathbb{P}_{\boldsymbol{F}\sim\mathscr{b}'_\star}[e \in \boldsymbol{F} \mid X \subseteq \boldsymbol{F}, Y \cap \boldsymbol{F} = \emptyset] \tag{9}$$

or add it to Y otherwise. In order to compute these probabilities, let $\mathcal{I}_{X,Y}(\vec{D}) = \{F \in \mathcal{I}(\vec{D}) \colon X \subseteq F \wedge Y \cap F = \emptyset\}$ denote all sets induced by a tuple $\vec{D}$ that contain X and are disjoint with Y. With this we can express Eq. (9) as $b_\star(\mathcal{I}_{X+e,Y}(\vec{D}))/b_\star(\mathcal{I}_{X,Y}(\vec{D}))$ for the case that $|X| \geq 2$. For the cases when the current solution X contains only one or no element the expression has to be modified slightly: in order to avoid generating singletons or the empty set, their $b_\star$-values have to be subtracted from the above expression.

Finally, it remains to compute the expression $b_\star(\mathcal{I}_{X,Y}(\vec{D}))$. This can be reduced to Equations (7) and (8) for the weight computation by constructing record tuples $\vec{D}_{X,Y}$ such that $\mathcal{I}_{X,Y}(\vec{D}) = \{F \cup X \colon F \in \mathcal{I}(\vec{D}_{X,Y})\}$:

$$\vec{D}_{X,Y}(i) = \begin{cases} \emptyset & \text{, if } i \in N \text{ and } X \not\subseteq \vec{D}(i) \\ \vec{D}(i) \setminus (X \cup Y) & \text{, otherwise} \end{cases}.$$

Then we can re-write $b_\star(\mathcal{I}_{X,Y}(\vec{D}))$ based on the choice of $\star$ as either $b_\Pi(\mathcal{I}_{X,Y}(\vec{D})) = b_\Pi(X) b_\Pi(\vec{D}_{X,Y})$ or $b_\Sigma(\mathcal{I}_{X,Y}(\vec{D})) = \left|\mathcal{I}(\vec{D}_{X,Y})\right| b_\Sigma(X) + b_\Sigma(\vec{D}_{X,Y})$.

4. LINEAR SPACE SAMPLING

In this section we present a CFTP-based implementation of Step 1 of the two-step sampling framework. After developing the algorithm, we show that it is applicable to a wide range of real-world datasets for pattern distributions based on 2, 3, and even 4 factors.

4.1 CFTP Algorithm

In order to design a CFTP algorithm for sampling tuples $\vec{\boldsymbol{D}} \sim w$ we have to construct a Markov chain on the data product $\mathfrak{D}$ with stationary distribution w. Moreover, this chain must have a corresponding random transition function that allows us to efficiently compute backward simulations and to detect coalescence (Eq. (4)). The right stationary distribution can be achieved by using the Metropolis-Hastings

Algorithm 3 CFTP Step 1

Require: i.i.d. $\vec{\boldsymbol{C}}_t, \boldsymbol{u}_t \sim \overline{w} \times u[[0,1)]$ with w as in (11),
Returns: tuple $\vec{\boldsymbol{D}} \sim w$ as in Step 1 of Alg. 1

1. $i \leftarrow 1$, $\vec{D} \leftarrow \perp$
2. **while** $\vec{D} = \perp$ **do**
3. $i \leftarrow i + 1$
4. **for** $t = 2^i, \ldots, 0$ **do**
5. **if** $\boldsymbol{u}_t \leq \frac{\overline{w}(\vec{D})}{w(\vec{D})} \frac{w(\vec{\boldsymbol{C}}_t)}{\overline{w}(\vec{\boldsymbol{C}}_t)}$ **then** $\vec{D} \leftarrow \vec{\boldsymbol{C}}_t$ *//with* $\frac{\overline{w}(\perp)}{w(\perp)} = 1$
6. **return** $\vec{D}$

construction (Eq. (2)) where on has the freedom of choosing a state proposal function. For this choice, we construct an upper bound of the weight function w, which results in a Markov chain that can be used efficiently within CFTP.

We start with the following state-transition specification (which is a simplified form of Eq. (2)): from a current state $\vec{D} \in \mathfrak{D}$ propose a new state $\vec{C} \in \mathfrak{D}$ based on some suitable proposal probability potential $q \colon \mathfrak{D} \to \mathbb{R}_+$ and accept this proposal if

$$\boldsymbol{u} \leq \frac{w(\vec{C}) q(\vec{D})}{w(\vec{D}) q(\vec{C})}$$

where $\boldsymbol{u} \sim u[[0,1)]$. That is, independently of the current state, we have a single probability function $q \colon \mathfrak{D} \to [0,1]$ for proposing the next state. Moreover, we will construct q such that it is strictly positive, i.e., $q(\vec{C}) > 0$ for all $\vec{C} \in \mathfrak{D}$. Besides guaranteeing ergodicity of the Markov chain, this allows efficient coalescence monitoring. In order to discuss this, consider the representation of the chain by a random transition function $\phi(\cdot, \boldsymbol{r})$—as specified in Eq. (3). With the above construction this is given naturally by letting $\boldsymbol{r} = (\vec{\boldsymbol{C}}, \boldsymbol{u})$ be a pair of a proposed state and a unit random real for checking acceptance. Suppose one knows a lower bound l on $\min_{\vec{D}\in\mathfrak{D}} q(\vec{D})/w(\vec{D})$. Then detecting coalescence, i.e., checking $|\mathrm{img}(\Phi_t)| = 1$, can be done by the implication

$$\boldsymbol{u}_i \leq l \frac{w(\vec{\boldsymbol{C}}_i)}{q(\vec{\boldsymbol{C}}_i)} \Rightarrow \left|\mathrm{img}(\phi(\cdot, (\vec{\boldsymbol{C}}_i, \boldsymbol{u}_i)))\right| = 1 \tag{10}$$

for $i \in \{t, \ldots, 1\}$, because the left-hand side implies that after applying $\phi(\vec{D}, (\vec{\boldsymbol{C}}_i, \boldsymbol{u}_i))$ all possible chain realizations coalesce to $\vec{\boldsymbol{C}}_i$ independent of their current state $\vec{D}$. A natural choice to achieve this is using the uniform distribution for proposal, i.e., to set $q = u[\mathfrak{D}]$. It can be efficiently simulated and finding the lower bound l then boils down to finding an upper bound to $\max_{\vec{D}\in\mathfrak{D}} w(\vec{D})$, which can be done efficiently for many relevant choices of the final pattern distribution $\mathcal{F}$.

However, one can improve this choice of the proposal probabilities in a way that—besides allowing efficient backward simulation and coalescence monitoring—also improves the expected coalescence time substantially. Suppose an upper bound to the weight function w is used for proposing new tuples in the CFTP algorithm, i.e., setting $q(\cdot) = \overline{w}(\cdot)$ with $\overline{w}(\vec{D}) \geq w(\vec{D})$ for all $\vec{D} \in \mathfrak{D}$. Then the lower bound l that is used in Eq. (10) becomes 1 by construction. On the one hand, we want an upper bound that is as tight as possible in order to maximize the coalescence probability—and consequently to minimize the computation time. On the other hand, we need a proposal function that we can efficiently compute and use for sampling. The following def-

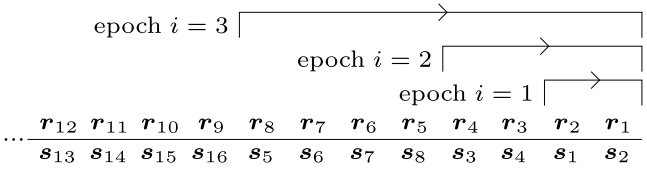


Figure 2: Order in which the elements of the infinite random string $(\boldsymbol{r}_i)_{i\in\mathbb{N}}$ are materialized by Alg. 3. The time horizon is doubled in every epoch. Only seed of random sequence $(\boldsymbol{s}_i)_{i\in\mathbb{N}}$ has to be stored to efficiently re-construct earlier materialized $\boldsymbol{r}_i$.

inition achieves the latter while at the same time provides good coalescence probabilities in theory and in practice (see Section 4.2):

$$\overline{w}(\vec{D}) = \prod_{i=1}^{c} \sqrt[c]{w_i(\vec{D}(i))} \tag{11}$$

where $w_i \colon \mathcal{D}_i \to \mathbb{R}_+$ defined by

$$w_i(D) = \begin{cases} b_\star(\mathcal{P}(D)) - b_\star(\overline{\mathcal{L}}(D)) & \text{, for } i \in P \\ b_\star(\mathcal{P}(E) \setminus \mathcal{P}(D)) - b_\star(\overline{\mathcal{L}}(D)) & \text{, for } i \in N \end{cases}$$

are component-wise weight functions. The w_i upper bound w in the following sense: for all tuples $\vec{D}$ with $\vec{D}(i) = D$ we have $w_i(D) \geq w(\vec{D})$. This follows from $b_\star$ being positive and $\mathcal{P}(\vec{D}(i)) \supseteq \mathcal{I}(\vec{D})$ for $i \in P$ as well as $\mathcal{P}(E) \setminus \mathcal{P}(\vec{D}(j)) \supseteq \mathcal{I}(\vec{D})$ for $j \in N$. Hence, we also have for their geometric mean $\overline{w}(\vec{D}) \geq w(\vec{D})$ as desired. Also, simulating $\vec{\boldsymbol{C}} \sim \overline{w}$ can be done efficiently by drawing $\vec{\boldsymbol{C}}(i)$ according to w_i independently per component. Implementing this enumeratively is not a problem, because the number of required weights are linear in the input and have to be computed only once.

This concludes the specification of our instantiation of CFTP. All ideas are summarized in the pseudo-code of Alg. 3 and in the proof of the following correctness statement.

Proposition 2. *With probability 1 Algorithm 3 terminates and returns a tuple drawn according to w.*

Proof. Suppose the algorithm terminates with final values of i and $\vec{D}$ equal to i^* and $\vec{D}^*$, respectively. In this case the if-condition in line 5 was true for some value t^* of t when $\vec{D} = \perp$. Then

$$\boldsymbol{u}_{t^*} \leq 1 \cdot \frac{w(\vec{\boldsymbol{C}}_{t^*})}{\overline{w}(\vec{\boldsymbol{C}}_{t^*})} \leq \frac{\overline{w}(\vec{B})}{w(\vec{B})} \frac{w(\vec{\boldsymbol{C}}_{t^*})}{\overline{w}(\vec{\boldsymbol{C}}_{t^*})}$$

for all $\vec{B} \in \mathfrak{D}$ because $\overline{w}$ is an upper bound to w, hence, $\overline{w}(\vec{B})/w(\vec{B}) \geq 1$. It follows that in this iteration Alg. 3 correctly computes $\mathrm{img}(\phi(\mathfrak{D}, (\vec{\boldsymbol{C}}_{t^*}, \boldsymbol{u}_{t^*})) = \{\vec{\boldsymbol{C}}_{t^*}\}$. For $t < t^*$ the inner loop correctly computes $\phi(\vec{D}, (\vec{\boldsymbol{C}}_t, \boldsymbol{u}_t))$. Thus, when the algorithm terminates it has correctly computed

$$\mathrm{img}(\Phi_{2^{i^*}}) = \\ \mathrm{img}(\phi(\cdot, (\vec{\boldsymbol{C}}_1, \boldsymbol{u}_1)) \circ \cdots \circ \phi(\cdot, (\vec{\boldsymbol{C}}_{t^*}, \boldsymbol{u}_{t^*}))) = \{\vec{D}^*\}$$

and by Eq. (4) we know that the output $\vec{D}^* \sim w$ as required.

That termination has a probability of 1 can be verified by checking that there is a tuple $\vec{B} \in \mathfrak{D}$ with $w(\vec{B}) > 0$ and consequently also $\overline{w}(\vec{B}) > 0$ (assuming $\mathcal{F}$ is a distribution). It follows that for every t there is a strictly positive probability of coalescence, hence, the probability of non-coalescence ($\vec{D} = \perp$) converges to zero for increasing t. □

Before we investigate the time requirements of Algorithm 3, we end this subsection with a remark clarifying that it can be implemented memory efficient. That is, it indeed leads to a linear space pattern sampling algorithm.

Remark 1. Regarding the *memory requirements*, it might seem that Algorithm 3 uses unbounded space because the realizations of the $\boldsymbol{r}_t = (\vec{\boldsymbol{C}}_t, \boldsymbol{u}_t)$ have to be reused in every epoch. However, assuming a standard model of randomness, this can be achieved with constant memory and without asymptotic time overhead. Assume the random sequence can be enumerated with unit delay between any two elements and constant memory from a single seed. The problem is that the random sequence is accessed in a reverse non-consecutive order: in epoch i the random variables $\boldsymbol{r}_{2^i}$ down through $\boldsymbol{r}_{2^{i-1}+1}$ are realized and then the realizations of the first 2^{i-1} variables are reused. Let $(\boldsymbol{s}_i)_{i\in\mathbb{N}}$ be the sequence we can generate consecutively. Applying a reordering scheme (see Fig. 2) we generate $\boldsymbol{r}_t$ based on $\boldsymbol{s}_{t'}$ with

$$t' = 3 \cdot 2^{\lceil \log_2 t \rceil - 1} - t + 1$$

and reconstruct $\boldsymbol{s}_t$ from $\boldsymbol{s}_1$ for t with $\lceil \log_2 t \rceil \neq \lceil \log_2 t - 1 \rceil$. The total reconstruction cost within a complete inner loop is $2t$—not effecting the asymptotic time complexity.

dataset	irreg	$\mathcal{F}_{\mathsf{frq}^3}$	$\mathcal{F}_{\mathsf{rr}\cdot\mathsf{frq}^2}$
pumsb	0	1	> 40.6
ionosphere	0	1	2.1
anneal	0	1	8.3
krvskp	0	1	53.4
hypothyroid	0.03	1.3	13.3
sick	0.03	1.4	15.4
censusincome2000	0.05	3.2	243.3
vote	0.06	1.8	2.6
icdmabstracts	0.23	> 8947848.9	> 8347048.2
mammals	0.28	> 53938.7	8716

Table 1: Reduction factor of simulation steps when using proposals w.r.t. $\overline{w}$ instead of uniform; higher irregularity (irreg) of dataset increases gain.

4.2 Time Complexity

For a theoretical analysis of the time complexity of Algorithm 3, note that the expected computation time is proportional to the expected time of the event that a complete coupling coalesces in one step. This time is geometrically distributed with a success probability

$$p_c = \mathbb{P}[w(\vec{\boldsymbol{C}})/\overline{w}(\vec{\boldsymbol{C}}) \geq \boldsymbol{u}] \ ,$$

hence the expected time is $1/p_c$. This probability, in turn, depends on how close $\overline{w}$ is to w, and, unfortunately, in general this can be exponentially small in the input size. However, in the special case of $\mathcal{D}_i = \mathcal{D}$ and $q_i = \mathsf{supp}$ for $i \in \{1, \ldots, c\}$, the worse case situation is much better; namely at most proportional to the inverse of the state space size $|\mathcal{D}|^c$. This setting is for instance met in all the distributions we consider in this paper except the discriminativity variants (but as we will see below, in practice the behavior of discrimininativity does not differ much from the theoretically good cases). The bound can be seen as follows: p_c is bounded from below by the probability that a tuple is proposed for which $\overline{w}$ is a sharp estimation of w, i.e., $\mathbb{P}[\overline{w}(\vec{\boldsymbol{C}}) = w(\vec{\boldsymbol{C}})] \leq p_c$. In particular this holds for the

tuples $\vec{D}$ with $\vec{D}(1) = \cdots = \vec{D}(c)$. Let $\vec{D}^*$ be one of these tuples maximizing w. Then $\vec{D}^*$ is proposed with probability

$$\overline{w}(\vec{D}^*) = \prod_{i=1}^{c} w_i(\vec{D}^*(i)) \tag{12}$$
$$= \prod_{i=1}^{c} \frac{\max_{D\in\mathcal{D}} b_\star(\mathcal{P}(D))}{\sum_{D\in\mathcal{D}} b_\star(\mathcal{P}(D))} \geq \prod_{i=1}^{c} \frac{1}{|\mathcal{D}|} = \frac{1}{|\mathcal{D}|^c}$$

as required. Thus, the order of the expected time is upper bounded by the pre-processing time $|\mathcal{D}|^c$ of the enumerative algorithm for drawing tuples—in other words: the CFTP variant can asymptotically not be worse than the straightforward implementation. It can, however, be much better. In contrast to the enumerative algorithm, CFTP is not bound to its worse case behavior. Instead it is adaptive to the hardness of the input dataset, which is why it is sometimes referred to as a "true algorithm" (e.g., [13]). Thus, ultimately the computation time "in practice" is the relevant quantity.

In order to assess the practical performance, we report the empirical[2] time requirements for 47 real-world datasets taken from the UCI machine learning repository [8] and from the FIMI repository [2] for adding some larger datasets. We test the CFTP algorithm against the enumerative solution for all distributions defined in Section 3.1 with $c \geq 2$ (for $c = 1$ the two algorithms are essentially identical). For having a realistic settings, we impose some memory constraint as well as some time constraint. The specific choices (here we use a 2GB memory cap and 10 minute time cap) carry relatively little significance for the baseline method: due to the preprocessing bottleneck, the enumerative baseline method has a fixed behavior in terms of both, time and space, that grows exponentially in c and is determined by the number of data records m. Hence, qualitatively we end up with the same results if we, e.g., double the limits (note that for the given settings it always hits the memory limit before it hits the time limit). The CFTP algorithm, on the other hand, is not affected by memory limits at all (as long as the dataset itself fits into main memory), and, although also affected by c and m can potentially behave much better than its worst case (m^c). What is more important is to fix a realistic number of patterns, because the enumerative method for Step 1 has to go through the preprocessing bottleneck only once, and then—given that it does not run out of memory—patterns are produced rapidly in time $\log^c m$ per pattern. Although the expected time of CFTP can be even smaller than $\log^c m$, the pattern number is still a sensitive quantity because it directly influences the baseline method's relative time per pattern—in contrast, for CFTP it is constant on expectation. For this purpose we use function (13) from the empirical evaluation of the pattern collections in application contexts (Section 5).

Table 2 contains a list of all datasets along with the number of data records, patterns, and the detailed results for the frequency distributions. The first observation is that, indeed, CFTP shows the anticipated good behavior: although the theoretical bound grows exponentially in the number of factors, even for $\mathcal{F}_{\mathsf{fq}^4}$ with $c = 4$ the computation can be performed within 1 seconds for 21 datasets, and for all but 3 datasets it can at least be performed within the 10 minute time limit. In contrast, the enumerative implementation of Step 1 runs out of memory already for small datasets as determined by m and c. The time requirements for the area variants, $\mathcal{F}_{\mathsf{area}}$ through $\mathcal{F}_{\mathsf{ar{\cdot}fq}^3}$, are essentially identical to the frequency variants. The reason for this is that they share the characteristic that the proposal weights $\overline{w}_i$ are only a function of the data record size. In case the dataset is (close to) being regular, this improves the coalescence time bound because all of the tuples of the form $\vec{D}(1) = \cdots = \vec{D}(c)$ contribute to the bound of Eq. 12. In contrast, the variants of $\mathcal{F}_{\mathsf{rare}}$ do not have this property, and, consequently for some datasets the behavior deviates upwards. All three groups, however, have the same abort statistics: while the enumerative baseline fails on 9 of the 47 datasets for 2-factor distributions, on 38 for $c = 3$, and on 46 for $c = 4$, the CFTP implementation fails on no dataset for $c = 2$, on 2 for $c = 3$, and on 3 for $c = 4$. Finally, although, not polynomially bounded, the variants of $\mathcal{F}_{\mathsf{dscr}}$ typically are even the least demanding. This is because $\mathfrak{D}$ is smaller for these distributions because

[2]Test system is Intel Core i5, 2.4GHz with Java 7 (Win7).

			$\mathcal{F}_{\mathsf{fq}^2}$		$\mathcal{F}_{\mathsf{fq}^3}$		$\mathcal{F}_{\mathsf{fq}^4}$	
dataset	#m	#pat	em	cp	em	cp	em	cp
labor	57	64	0	0	0	0	12	1
zoo	101	119	0	0	1	0	-	0
lymph	148	136	0	0	4	0	-	1
iris	150	36	0	0	3	0	-	0
hepatitis	155	130	0	0	5	1	-	2
wine	178	104	0	0	6	1	-	20
autos	205	191	0	0	12	5	-	214
glass	214	77	0	0	11	0	-	0
audiology	226	531	0	0	17	1	-	6
heartstatlog	270	113	0	0	-	0	-	1
breastcancer	286	73	0	0	-	0	-	0
hearth	294	90	0	0	-	0	-	0
heartc	303	107	0	0	-	0	-	2
primarytumor	339	142	0	0	-	0	-	1
ionosphere	351	295	0	0	-	26	-	-
colic	366	144	0	0	-	7	-	82
vote	435	140	0	0	-	0	-	0
balancescale	625	46	0	0	-	0	-	0
soybean	683	301	1	0	-	4	-	44
credita	690	141	1	0	-	1	-	4
breastw	699	85	0	0	-	0	-	0
diabetes	768	86	1	0	-	0	-	1
vehicle	846	184	1	0	-	5	-	135
icdmabstracts	859	477	1	0	-	0	-	0
annealsmall	898	137	1	0	-	0	-	1
anneal	898	382	1	0	-	1	-	2
tictactoe	958	99	1	0	-	0	-	1
vowel	990	139	1	0	-	2	-	32
creditg	1000	209	1	0	-	4	-	22
germanstatlog	1000	209	1	0	-	4	-	21
pokerh2000	2000	120	5	0	-	1	-	10
censusinc2000	2000	427	5	0	-	6	-	98
mammals [15]	2183	266	6	0	-	1	-	16
segment	2310	223	7	0	-	3	-	37
krvskp	3196	430	15	0	-	2	-	26
hypothyroid	3772	332	19	0	-	1	-	1
sick	3772	332	19	0	-	1	-	1
abalone	4177	108	22	0	-	0	-	0
mushroom	8124	285	-	0	-	3	-	25
censusinc10K	10K	518	-	1	-	9	-	141
pendigits	11K	228	-	1	-	6	-	90
nursery	13K	122	-	0	-	0	-	1
letter	20K	242	-	1	-	29	-	421
adult	49K	218	-	1	-	1	-	1
pumsb	49K	1153	-	28	-	-	-	-
connect	68K	689	-	3	-	36	-	256
accidents	340K	606	-	43	-	-	-	-

Table 2: Datasets along with detailed computation time for the frequency distributions; "-" corresponds to out of time (>10 minutes) for CFTP or out of memory (>2GB) for enumerative; in case of CFTP the median of five repetitions is reported.

the $\mathcal{D}_l$ are proper subsets of $\mathcal{D}$, hence, the individual factors are smaller. The corresponding CFTP abort statistics are 0, 1, and 4 for 2, 3, and 4 factors, respectively; as opposed to 5, 28, and 46 for the enumerative baseline. In summary, the CFTP technique substantially increases the set of distributions and datasets for which two-step sampling is applicable.

We close this investigation with a note on the effectiveness of choosing $\overline{w}$ over uniform as proposal probabilities. Table 1 contains the reduction factor of simulated time steps for a few representative datasets. Again, the behavior differs between the $\mathcal{F}_{\mathsf{rare}}$ variants and, e.g., the $\mathcal{F}_{\mathsf{frq}}$ variants. Here the reason is that the closer the dataset is to being regular (i.e., $|D| = |D'|$ all $D, D' \in \mathcal{D}$) the closer is $\overline{w}$ to u for those distributions where $\overline{w}_i$ is a function of the record size. Consequently, while the gain of $\overline{w}$ over u is substantial throughout all datasets for rare, for the other distributions this depends on the degree of regularity. Irregularity is measured here by $\big||D| - \mathrm{avg}_{D\in\mathcal{D}}\,|D|\big| \,/ \|\mathcal{D}\|$.

5. EVALUATION OF DISTRIBUTIONS

In the previous sections we developed an algorithm that can be used to efficiently perform pattern sampling based on three or even four factor distributions. Now we show that using such distributions brings indeed a substantial boost in pattern quality.

5.1 Unsupervised Performance

In order to assess the unsupervised descriptive performance of a pattern collection we use the pattern-based compressor Krimp [19]. Krimp selects a sub-collection of the extracted patterns and builds a code table from them along with a compressed version of the input dataset. The code table assigns code symbols to the patterns it contains, and occurrences of the patterns are replaced by their respective code symbols in the compressed database. Krimp approximates a minimum description length code table, i.e., one that minimizes the combined length of the code table and the compressed dataset. Note that this principle factors in redundancy—a pattern is not selected for the code table if it does not achieve sufficient additional compression (over the other selected patterns) that compensates for its own size. The size reduction achieved this way can be regarded as a measure of the information about the dataset that is contained in a pattern collection.

	c	avg r	r1	rq	rh	avg s
area · frq^2	3	3.70	9	23	37	0.767
frq^3	3	4.48	2	16	38	0.768
rare · frq^3	4	4.52	15	22	29	0.768
area · frq^3	4	4.77	1	17	34	0.767
frq^4	4	5.32	2	17	29	0.768
rare · frq^2	3	6.14	6	13	25	0.797
frq^2	2	7.05	1	4	16	0.785
frq	1	7.45	2	4	14	0.787
rare · frq	2	7.48	0	4	14	0.800
area · frq	2	7.75	2	6	15	0.790
top-k closed	–	8.61	4	5	11	0.824
area	1	11.50	0	1	1	0.859
rare	1	11.95	0	1	1	0.890

Table 3: Pattern extraction methods ordered by average rank (avg r) in Krimp experiments; number of factors (c), number of datasets with top rank (r1), upper quarter rank (rq), upper half rank (rh), and average score, i.e., relative size reduction (avg s).

In our experiment we use all datasets from Table 2, and for each we consider pattern collections consisting of size

$$k(\mathcal{D}) = \log|\mathcal{D}| \operatorname*{avg}_{D\in\mathcal{D}} |D| \quad . \tag{13}$$

We include all distributions from Section 3.1 that do not rely on label information. In addition we include top-k frequent closed set extraction as a deterministic pattern collection that can be extracted efficiently (see [17]), i.e., in polynomial time of the combined size of the input dataset and the output pattern collection. The same could be done with the top-k frequent patterns (including non-closed), but closed patterns are a stronger baseline for moderate values of k. Thus, all methods considered here are true polynomial (expected) time algorithms.

The results are summarized in Table 3 where all methods are listed ordered according to the average rank they achieved on all datasets. It stands out firmly that the distribution with $c = 3$ and $c = 4$ are dominating the ranking. A notable difference between the family of distributions based on rare singletons to the others is that it stills receives a substantial gain in the $c = 4$ variant over the $c = 3$ variant. In contrast, for the families based on plain frequency and area, the fourth factor does not improve the average rank anymore. For some datasets it is even counter-productive. It is not surprising that the effectiveness of adding powers of frequency to the distributions has a natural limit, because at a certain point this will degenerate the pattern collections again to sets of high frequency and high redundancy patterns. Finally, it can be observed that already the distributions with $c = 2$ outperform the deterministic baseline of the top-k closed frequent sets.

5.2 Supervised Performance

	c	avg r	r1	rq	rh	avg s
dscr · frq$^2_+$	4	4.38	19	28	34	2.118
rare · frq^2	3	4.63	6	19	36	1.734
dscr · frq$_+$	3	4.8	5	23	36	1.966
frq^3	3	7	0	8	30	1.772
rare · frq	2	7.15	2	11	26	1.546
rare · frq^3	4	7.33	3	15	27	1.589
dscr · frq	3	7.58	1	11	27	1.771
dscr · frq^2	4	7.7	0	12	30	1.844
area · frq^2	3	8.53	0	6	22	1.663
frq^4	4	8.98	0	6	20	1.689
area · frq^3	4	8.98	0	6	20	1.644
frq^2	2	10.18	0	3	13	1.406
area · frq	2	11.68	0	0	8	1.298
top-k closed	–	12.2	3	5	12	1.024
dscr	2	13.95	0	0	3	0.829
rare	1	14.1	0	1	4	0.788
frq	1	15.7	0	1	1	0.667
area	1	16.18	1	1	2	0.646

Table 4: Pattern extraction methods ordered by their average rank (avg r) in the information gain experiments with all attributes as in Table 3.

In a second experiment, we evaluate the supervised performance of the random pattern collections. Patterns that help distinguish between different classes, are needed in a variety of application contexts, ranging from exploratory data analysis to pattern-based-classification. In exploratory data analysis a domain expert considers the patterns in their own right, e.g., for finding malfunctioning or high-value subgroups of a population. In pattern-based-classification the

patterns are human-readable building blocks of a compound prediction model [14]. In both cases, redundant information about the target label, i.e., information that is already contained in other result patterns, is useless or even counterproductive. Therefore, we settle here for an evaluation measure that is based on the pattern's information gain, downweighted by its similarity to more informative patterns. The score of a pattern collection $\mathcal{F} = \{F_1, \ldots, F_k\}$, where the F_i are in descending order of information gain, is defined as

$$s(\mathcal{F}) = \mathsf{ig}(F_1) + \sum_{i=2}^{k} \mathsf{ig}(F_i) \min_{j \in \{1,\ldots,i\}} \delta(F_j, F_i)$$

where $\delta(F_i, F_j)$ is the Jaccard distance between the extension of the patterns

$$\delta(F_i, F_j) = 1 - |\mathcal{D}[F_i] \cap \mathcal{D}[F_j]| \,/\, |\mathcal{D}[F_i] \cup \mathcal{D}[F_j]| \quad .$$

When used for feature selection, this measure leads to high accuracy pattern-based classifiers [6].

The results are summarized in Table 4; this time also including the distributions from the discriminativity group. Again, all methods are ordered according to the average rank achieved on all datasets. Similarly to the unsupervised performance, also in terms of supervised performance, distributions with $c = 3$ and $c = 4$ dominate the ranking with the only exception of $\mathsf{rare} \cdot \mathsf{frq}$. A further similarity is that variants with four factors do not necessarily improve over their three factor counterparts. An important exception is the overall best method $\mathsf{dscr} \cdot \mathsf{frq}_+^2$. In this case, the additional factor further emphasizes the difference in the support of the patterns per class, hence it is chosen sensitively to the supervised evaluation scenario. Finally, note that top-k frequent closed sets are again outperformed by most of the random collections.

6. DISCUSSION

The CFTP-based variant of the two-step sampling framework that is developed in this paper avoids the super-linear space requirement of the previous enumerative version of Step 1. In addition, the expected running time of CFTP is usually much smaller than the previously fixed pre-processing time, and for many pattern distributions it is also theoretically upper bounded by the complexity of the enumerative Step 1. Consequently, the new algorithm allows to efficiently draw samples for previously intractable distributions containing up to four factors. The experimental evaluation of these distributions shows that going up from two to three or four factors, indeed, boosts the pattern quality substantially.

At the same time one can also observe a satiation effect in the sense that the step from three to four factors does not provide as much gain as the step from two to three. Therefore, for future research it appears to be of higher priority to identify the distributions best suited for specific application tasks than to further boost the number of possible factors. In particular, stratified random pattern collections that are drawn from a mixture of distributions (and possible deterministic methods) appear to be worth investigating. Also, a currently unused potential lies in post-processing the random pattern collections by applying fast local optimization procedures. Finally, it is open how to lift the two-step sampling framework to large scale inputs that do not fit into main memory.

Acknowledgment. This work was supported by the DFG (GA 1615/2-1 and GA 1615/1-1), the EC (ICT-FP7-LIFT-255951) and Research Foundation Flanders (FWO).

7. REFERENCES

[1] M. Al Hasan and M. J. Zaki. Output space sampling for graph patterns. *PVLDB*, 2(1):730–741, 2009.

[2] R. Bayardo, B. Goethals, and M. J. Zaki, editors. *IEEE ICDM Workshop on Frequent Itemset Mining Implementations, 2004*, volume 126 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2004.

[3] M. Boley, T. Gärtner, and H. Grosskreutz. Formal concept sampling for counting and threshold-free local pattern mining. In *SDM*, pages 177–188, 2010.

[4] M. Boley, C. Lucchese, D. Paurat, and T. Gärtner. Direct local pattern sampling by efficient two-step random procedures. In *KDD*, pages 582–590, 2011.

[5] V. Chaoji, M. A. Hasan, S. Salem, J. Besson, and M. J. Zaki. Origami: A novel and effective approach for mining representative orthogonal graph patterns. *Stat. Anal. and Data Min.*, 1(2):67–84, 2008.

[6] H. Cheng, X. Yan, J. Han, and C.-W. Hsu. Discriminative frequent pattern analysis for effective classification. In *ICDE*, pages 716–725, 2007.

[7] G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. In *KDD*, pages 43–52. ACM, 1999.

[8] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[9] F. Geerts, B. Goethals, and T. Mielikäinen. Tiling databases. In *DS*, pages 278–289. Springer, 2004.

[10] H. Grosskreutz, S. Rüping, and S. Wrobel. Tight optimistic estimates for fast subgroup discovery. In *ECML/PKDD, Part I*, pages 440–456, 2008.

[11] D. J. Hand. Pattern detection and discovery. In *ESF Exploratory Workshop on Pattern Detection and Discovery*, pages 1–12. Springer, 2002.

[12] W. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[13] M. Huber. Perfect sampling using bounding chains. *Annals of App. Prob.*, 14(2):734–753, 2004.

[14] A. J. Knobbe, B. Crémilleux, J. Fürnkranz, and M. Scholz. From local patterns to global models: the lego approach to data mining. In *From Local Patterns to Global Models: Proceedings of the ECML/PKDD 2008 Workshop*, 2008.

[15] A. Mitchell-Jones. *The Atlas of European Mammals*. Poyser Natural History. T & AD Poyser.

[16] S. Morishita and J. Sese. Traversing itemset lattice with statistical metric pruning. In *PODS*, pages 226–236, 2000.

[17] A. Pietracaprina and F. Vandin. Efficient incremental mining of top-k frequent closed itemsets. In *DS*, pages 275–280, 2007.

[18] J. G. Propp and D. B. Wilson. Exact sampling with coupled markov chains and applications to statistical mechanics. *Rand. Struct. Alg.*, 9(1-2):223–252, 1996.

[19] J. Vreeken, M. van Leeuwen, and A. Siebes. Krimp: mining itemsets that compress. *Data Min. Knowl. Discov.*, 23(1):169–214, 2011.