An Efficient Approximation of Concept Stability Using Low-Discrepancy Sampling

Mohamed-Hamza Ibrahim(✉) and Rokia Missaoui

Département d'informatique et d'ingénierie, Université du Québec en Outaouais, 101, rue St-Jean Bosco, Gatineau, Québec J8X 3X7, Canada
{ibrm05,rokia.missaoui}@uqo.ca

Abstract. One key challenge in Formal Concept Analysis is the scalable and accurate computation of stability index as a means to identify relevant formal concepts. Unfortunately, most exact methods for computing stability have an algorithmic complexity that could be exponential w.r.t. the context size. While randomized approximate algorithms, such as Monte Carlo Sampling (MCS), can be good solutions in some situations, they frequently lead to the slow convergence problem with an inaccurate estimation of stability. In this paper, we introduce a new approximation method to estimate the stability using the low-discrepancy sampling (LDS) approach. To improve the convergence rate, LDS uses quasi-random sequence to distribute the sample points evenly across the power set of the concept intent (or extent). This helps avoid the clumping of samples and let all the areas of the sample space be duly represented. Our experiments on several formal contexts show that LDS can achieve faster convergence rate and better accuracy than MCS.

Keywords: Formal Concept Analysis · Concept stability
Low-discrepancy sampling · Pattern relevancy

1 Introduction

Formal Concept Analysis (FCA) [10] is a theoretical framework based on lattice and order theory [10], which is a formalization of concept lattice and hierarchy. The concept lattice contains frequently substantial combinatorial structures that are exploited for data mining and analysis purposes. However, a large amount of such structures could be irrelevant, which in turn, potentially induce a high complexity even for small datasets [4,11]. Thus, an important challenge in the big data era is to discover only relevant concepts from a possibly very large and complex lattice. Inspired by the FCA theory,, concept selection techniques are commonly used to focus on only relevant parts of a concept lattice [16,17]. The basic idea is to single out a subset of important concepts, objects or attributes based on relevance measures. Several selection measures have been introduced as

© Springer International Publishing AG, part of Springer Nature 2018
P. Chapman et al. (Eds.): ICCS 2018, LNAI 10872, pp. 24–38, 2018.
https://doi.org/10.1007/978-3-319-91379-7_3

clearly detailed in [17], such as stability [13,15], separation [11], probability [16] and robustness [11]. While each selection measure exploits certain valuable characteristics in the formal context, stability has been found to be more prominent for assessing concept relevancy [16].

The stability index of the concept $c = (A, B)$ quantifies how its intent B depends on the set of objects in its extent A. Thus, the stability frequently provides a robust indication of how much noise appears in the concept. Unfortunately, computing the stability is #P-complete [2,15] approximating, and we typically need to compute the set of all generators associated with the concept to get its exact stability value. This could be problematic with large-sized concepts, and thus, a practical solution to this problem could be the stability approximation. The authors in [2] introduced random MCS as a way to approximate the stability. Although MCS might give a good approximation in some given cases [18], it converges very slowly and needs more samples to reduce the sampling error. This is due to the fact that MCS randomly chooses samples independently. This leads MCS to end up with some regions in the power set space of the concept intent (or extent) with too many samples tightly clumped, while other regions have no samples. This weakens the convergence rate[1] of the sampling process and worsen the inaccuracy of stability [21,22]. In this paper, our main objective is to use a low-discrepancy approach [7,24] to address the limitation of MCS.

In the following we first give a background, then we explain our general LDS approach in Sect. 3 and illustrate its application to stability approximation. Finally, we present experimental evaluations in Sect. 4, followed by our conclusion in Sect. 5.

2 Background

2.1 Formal Concept Analysis

FCA is a mathematical formalism for data analysis [10] that uses a formal context as input to construct a set of formal concepts organized in a concept lattice. A *formal context* is a triple $\mathbb{K} = (\mathcal{G}, \mathcal{M}, \mathcal{I})$, where $\mathcal{G}$ is a set of objects, $\mathcal{M}$ a set of attributes, and $\mathcal{I}$ a relation between $\mathcal{G}$ and $\mathcal{M}$ with $\mathcal{I} \subseteq \mathcal{G} \times \mathcal{M}$. For $g \in \mathcal{G}$ and $m \in \mathcal{M}$, $(g, m) \in \mathcal{I}$ holds iff the object g has the attribute m. Given arbitrary subsets $A \subseteq \mathcal{G}$ and $B \subseteq \mathcal{M}$, the Galois connection can be defined by the following derivation operators:

$$A' = \{m \in \mathcal{M} \mid \forall g \in A, (g, m) \in \mathcal{I}\},\ A \subseteq \mathcal{G}$$

$$B' = \{g \in \mathcal{G} \mid \forall m \in B, (g, m) \in \mathcal{I}\},\ B \subseteq \mathcal{M}$$

where A' is the set of attributes common to all objects of A and B' is the set of objects sharing all attributes from B. The closure operator $(.)''$ implies the double application of $(.)'$, which is extensive, idempotent and monotone. The

[1] The convergence rate quantifies how quickly the sampling error decreases with an increase in the number of samples.

subsets A and B are said to be closed when $A'' = A$, and $B'' = B$. If both A and B are closed and $A' = B$, and $B' = A$, then the pair $c = (A, B)$ is called a *formal concept* of $\mathbb{K}$ with *extent* A and *intent* B. For a finite intent (or extent) set of w elements, we use $\mathcal{P}(.)$ to denote its power set with a number of subsets equals to $n = 2^w$, i.e., the set of all its subsets, including the empty set and the set itself.

2.2 Stability Index

The stability index $\sigma(c)$ is an interestingness measure of a formal concept c for selecting relevant patterns [5,11,15,23].

Definition 1. *Let $\mathbb{K} = (\mathcal{G}, \mathcal{M}, \mathcal{I})$ be a formal context and $c = (A, B)$ a formal concept of $\mathbb{K}$. The intensional stability $\sigma_{in}(c)$ can be computed as [2,5]:*

$$\sigma_{in}(c) = \frac{\mid \{e \in \mathcal{P}(A) | e' = B\} \mid}{2^{|A|}} \tag{1}$$

In a dual way, the extensional stability $\sigma_{ex}(c)$ is defined as:

$$\sigma_{ex}(c) = \frac{\mid \{e \in \mathcal{P}(B) | e' = A\} \mid}{2^{|B|}} \tag{2}$$

In Eq. (1), intensional stability $\sigma_{\text{in}}(c)$ measures the strength of dependency between the intent B and the objects of the extent A. More precisely, it expresses the probability to maintain B closed when a subset of noisy objects in A are deleted with equal probability. The numerator of Eq. (1) can be calculated by naively iterating through all the subsets in $\mathcal{P}(A)$ and counting the number of those subsets whose derivation operator produces the intent B. This computation needs a time complexity of $O(2^{|A|})$. However, it had been shown in [23] and later on in [25] that the numerator of $\sigma_{\text{in}}(c)$ in Eq. (1) can be computed by identifying and counting the minimal generators of the concept. Such computation takes a time complexity of $O(L^2)$ [23,25], where L is the size of the concept lattice, and requires the lattice construction which needs a time complexity of $O(|\mathcal{G}|^2 \cdot |\mathcal{M}| \cdot L)$ [14,20]. In a dual way, the extensional stability $\sigma_{\text{ex}}(c)$ (see Eq. (2)) measures overfitting in the intent B. When it is high for the extent A of a given concept c, it means that A would stay closed even when we discard noisy attributes from its intent B. Similar to the intensional stability, $\sigma_{\text{ex}}(c)$ can be computed by simply iterating through all the subsets in $\mathcal{P}(B)$ but counting the number of those subsets for which the derivation operator produces the extent A. This computation also needs exponential time complexity $O(2^{|B|})$.[2]

[2] Note that since both $\sigma_{\text{in}}(c)$ and $\sigma_{\text{ex}}(c)$ provide dual measurements, we generally use $\sigma(c)$ throughout the rest of the paper.

2.3 Approximating Stability

With a large-sized intent (or extent), the exponential time complexity of computing the stability represents a bottleneck for any exact method. For instance, to exactly compute $\sigma(c)$ of a concept with an intent size $n = 21$, we need to perform more than two million computational comparisons. Thus, a practical solution is to approximate the stability using MCS method [2,6] as shown in Algorithm 1. Under a certain number of samples N, MCS iteratively picks up a random subset from the intent power set $\mathcal{P}(B)$ and increments a counter when the picked up subset satisfies the condition in the numerator of Eq. (2) (see lines 3–5). It ultimately uses the counter to estimate the stability (see line 8). However, how many samples are needed in the MCS to accurately estimate the stability? In [2,5] the authors demonstrated that at least a sample size of $N > \frac{1}{2\epsilon^2} \ln \frac{2}{\delta}$ could be sufficient for MCS to estimate the stability with precision ϵ and error rate δ. This means that, with a probability $1 - \delta$, we can calculate an approximation $\tilde{\sigma}(c)$ of the exact stability $\sigma(c)$ within the interval $\left[\tilde{\sigma}(c) - \epsilon, \tilde{\sigma}(c) + \epsilon\right]$. At a first sight, this appears to be a low-cost computational time compared to the exact computational one, but in fact it is not. For example, MCS needs a sample size $N > 3.8 \times 10^6$ to accurately estimate the stability with precision $\epsilon = 0.001$ and error rate $\delta = 0.001$.

Algorithm 1. MCS for stability approximation

```
Input: Concept $c = (A, B)$ and Sample size $N$.
Output: Estimated stability $\tilde{\sigma}(c)$.
1: Count ← 0;
2: for $i \leftarrow 1$ to $N$ do
3:     Pick up a random subset $e$ in $\mathcal{P}(B)$;
4:     if $e' == A$ then
5:         Count ← Count + 1;
6:     end if
7: end for
8: $\tilde{\sigma}(c) \leftarrow \frac{\text{Count}}{N}$;
```

3 LDS for Stability

In this section we will explain the usage of the LDS method for estimating stability.

3.1 LDS Framework

Let us first reformulate the problem of computing stability so that the LDS sampling can be easily comprehended. Given a formal concept c with extent A and intent B, then the power set of B, $\mathcal{P}(B) = \{b_{x_1}, \ldots, b_{x_n}\}$ is associated with a set of indices $X = \{x_1, \ldots, x_n\}$, where $x_i \in X$ is the index of $b_{x_i} \in \mathcal{P}(B)$. We define $\mathbb{1}(x)$ as an indicator function over X as follows:

$$\mathbb{1}(x) = \begin{cases} 1 & \text{If } b'_x = A,\ b_x \in \mathcal{P}(B) \\ 0 & \text{Otherwise} \end{cases} \tag{3}$$

where $\mathbb{1}(x)$ is equal to 1 when a randomly picked up subset $b_x \in \mathcal{P}(B)$ satisfies the condition in the numerator of stability in Eq. (2); and 0 otherwise. Note that the concept stability in Eq. (2) is identical to the following quantity:

$$\sigma(c) \equiv p(b'_x = A \mid b_x \in \mathcal{P}(B)) \tag{4}$$

where $p(b'_x = A \mid b_x \in \mathcal{P}(B))$ is the probability of randomly picking up a subset b_x and finding out that it satisfies the stability condition. This probability can be computed by using the expectation $E[.]$ of the indicator function $\mathbb{1}(x)$ in Eq. (3) as follows:

$$p(b'_x = A \mid b_x \in \mathcal{P}(B)) = E[\mathbb{1}(x)] = \sum_{x \in X} \mathbb{1}(x) P_X(x) \tag{5}$$

where $P_X(x)$ is the (univariate) probability mass function of X. If we assume that the subsets in $\mathcal{P}(B)$ are uniformly distributed, then

$$P_X(x) = \frac{1}{n},\ \forall x \in X \tag{6}$$

where $n = |X|$ is the size of the power set. Now if we use Eqs. (4) and (6) into (5), then the stability in Eq. (2) can be computed as:

$$\sigma(c) = E[\mathbb{1}(x)] = \frac{1}{n} \sum_{x \in X} \mathbb{1}(x) \tag{7}$$

The stability in Eq. (7) can be approximated by taking a set of samples $\mathcal{S}$ as:

$$\sigma(c) \approx \tilde{\sigma}(c; N) = \frac{1}{N} \sum_{x_s \in \mathcal{S}} \mathbb{1}(x_s) \tag{8}$$

where $N = |\mathcal{S}|$ is the sample size. Now, how can we select correlated N samples (i.e., subsets), in Eq. (8), that are distributed across the power set space more uniformly than uncorrelated random samples in MCS? The answer is the usage of low-discrepancy sequence. In the following subsection, let us explain how to generate low-discrepancy (also called quasi-random) sample points.

3.2 Generating Low-Discrepancy Sequences

Various deterministic methods can be typically used to generate low-discrepancy sequences of N sample points $\{s_i\}_{i=1}^{N}$. Here we focus on the two commonly used sequences called *Scrambled Van der Corput* [9,24] and *Sobol* [12,19].

(I) Scrambled Van der Corput Sequence (VDC) is the simplest low discrepancy sequence. Starting with a prime integer $r \geq 2$ as a radical base to represent the sequence, we can obtain the i$^{\text{th}}$ sample point s_i by calling the function `GenerateSVdC(i,r)` in Algorithm 2. We first represent the decimal number i in radical base r (line 1). Then, we put a radical point in the front of its reversal representation and convert it back to the decimal (Line 2). We finally scramble the result by shifting it with a pseudo-random number (see lines 2–4). For example, if we choose base $r = 2$ in Scrambled VDC, then we can get the 4th sample point s_4 as follows: (1) express $i = 4$ in base 2 as $(100)_2$; (2) reverse the digits and put a radix point in front of it to get $(0.001)_2$; (3) convert it back to a decimal number to get $h_4 = 0.375$; (4) scramble h_4 with any random number (say $d_4 = 0.8$) to obtain $s_4 = 0.375 + 0.8 \pmod 1 = 0.175$.

Algorithm 2. Generating a sample point in Scrambled VDC.

```
Function GenerateSVdC
Input: Number $i$ and radical base $r$
Output: A point $s_i \in [0,1]$
       // Express $i$ as a number in base $r$.
    1: $i \rightarrow \sum_{j=0}^{l} a_j r^j$;
       // Reverse the digits, put a radix point and convert it back to
       decimal.
    2: $h_i \leftarrow \sum_{j=0}^{l} a_j r^{-j-1}$;    // $l = \lfloor \log_r i \rfloor$ .
       // Scramble the number by randomized shifting.
    3: $d_i \leftarrow$ Generate a uniform random number $\in [0,1]$;
    4: $s_i \leftarrow h_i + d_i \pmod 1$;
    5: Return $s_i$;
```

(II) One-dimensional Sobol Sequence is often generated based on radical base $r = 2$. Unlike Scrambled VDC with base 2, it applies the permutations based on a set of direction numbers (instead of reversing numbers). The function `GenerateSobol(i)` in Algorithm 3 summarizes the pseudo-code of Sobol for generating a sample point s_i. We first compute the gray-code h_i of the number i by using the bit-by-bit exclusive-or operator (see line 1). Then we transform h_i to its binary representation (line 2). Subsequently, we sum up bit by bit exclusive-or of the direction numbers associated with the digits of h_i that are different from zero (see line 3). Note that the set of direction numbers ($\{v_j = \frac{z_j}{2^j}\}$, where $0 < z_j < 2^j$) is a sequence of binary fractions with bits after the binary point. Thus, they can be uniquely defined as $v_1 = (0.1)_2$, $v_2 = (0.11)_2$, $v_3 = (0.111)_2$, and so on. Now, as a concrete example, to obtain the 2nd sample point s_2 in the Sobol sequence, we do the following: (1) compute the gray code of $i = 2$ as $h_2 = 2 \oplus 1 = (10)_2 \oplus (1)_2 = (11)_2$; (2) apply exclusive-or to the first $v_1 = 0.1$ and the second $v_2 = 0.11$ direction numbers to get $u_2 = 0.1 \oplus 0.11 = 0.01$; (3) convert u_2 back to decimal to obtain Sobol point $s_2 = (0.01)_2 = 0.25$.

The limitation of $q-$dimensional Sobol sequence is that its convergence rate $O(\frac{(\log N)^q}{N})$ is smaller than $O(\frac{1}{\sqrt{N}})$ of the MCS convergence rate only when the problem dimension q is very small [19]. However, this limitation does not occur in our algorithms because the power set $\mathcal{P}(B)$ of an intent B has often one

Algorithm 3. Generating a sample point in Sobol sequence.

```
Function GenerateSobol
Input: Number $i$
Output: A point $s_i \in [0,1]$
    // Compute the Gray Code of number $i$.
 1: $h_i \leftarrow i \oplus \lceil \frac{i}{2} \rceil$ ;
    // Express $h_i$ as a number in base 2.
 2: $h_i \rightarrow \sum_{j=0}^{l} a_j 2^j$;    // $l = \lfloor \log_r i \rfloor$ .
    // Sum up the (XOR) of direction numbers associated with the digits of
    $h_i$.
 3: $u_i \leftarrow a_1 v_1 \oplus \ldots \oplus a_l v_l$;
    // Put a radix point in the front of $u_i$ and convert it back to
    decimal.
 4: $s_i \leftarrow \sum_{j=0}^{l} a_j 2^{-j-1} \leftarrow u_i$;
 5: Return $s_i$;
```

dimension (i.e., $q = 1$). Now, let us continue back to the approximation of stability in the next subsection.

3.3 LDS Algorithm for Estimating Stability

Here our goal is to improve the convergence rate by using LDS approach.

Algorithm 4 is the pseudo-code of the LDS procedure for approximating the stability depicted in Eq. (8). The algorithm takes as input the number of samples and the radical base, and then it applies two steps. In a first stage (see lines 1–6), LDS generates a low-discrepancy sequence $\mathcal{S}$ by using either the Scrambled VDC method (i.e., calling `GenerateSVdC(i,r)` in Algorithm 2) or Sobol method (i.e., calling `GenerateSobol(i)` in Algorithm 3). In a second stage, LDS exploits the obtained quasi-random sequence $\mathcal{S}$ to uniformly select correlated subsets across all areas of the intent power-set. That is, LDS iteratively uses each sample point $s_i \in \mathcal{S}$ to pick up a corresponding subset index x_i by computing the inverse cumulative function $\mathcal{F}^{-1}(s_i)$ (see lines 9–10). Here, it is worth noting that because X is uniformly distributed (refer to Eq. (6)), then it has a discrete uniform cumulative distribution function $\mathcal{F}(X)$, in which its inverse $\mathcal{F}^{-1}()$ can be calculated as follows:

$$x_i = \mathcal{F}^{-1}(s_i) = \lceil s_i \times n \rceil \tag{9}$$

Then, it uses the indicator function (see Eq. (6)) to check the stability condition of the selected subset, and it consequently increments the *count* whenever the condition is held (see lines 11–14). Thus, *count* often captures the number of these selected subsets that satisfy the condition in the numerator of Eq. (8). Finally, the stability can be estimated as the portion of those subsets, from the N picked up ones, that satisfy the stability condition (line 15).

Algorithm 4. LDS for stability approximation.

```
Input: Concept $c = (A, B)$, sample size $N$ and base $r$
Output: Stability $\tilde{\sigma}(c)$.
    // Stage 1: Generate low-discrepancy sequence of points
 1: $\mathcal{S} \leftarrow \emptyset$;
 2: for $i \leftarrow 1$ to $N$ do
 3:     $s_i \leftarrow$ GenerateSVdC(i,r);  // Case 1: using Scrambled VDC in Algo. 2.
    // OR
 4:     $s_i \leftarrow$ GenerateSobol(i);   // Case 2: using sobol in Algo. 3.
 5:     $\mathcal{S} \leftarrow \mathcal{S} \cup \{s_i\}$;
 6: end for
    // Stage 2: Approximate the stability
 7: Count $\leftarrow 0$;
 8: for each $s_i$ in $\mathcal{S}$ do
    // pick up a subset using the inverse cdf $\mathcal{F}^{-1}(.)$
 9:     $x_i \leftarrow \lceil s_i \times n \rceil$;
10:     $b_{x_i} \leftarrow$ Subset in $\mathcal{P}(B)$ at index $x_i$;
    // Use $\mathbb{1}(.)$ in Eq. (3) to check if the subset satisfies the stability
    condition
11:     if $\mathbb{1}(b_{x_i}) == 1$ then
12:         Count $\leftarrow$ Count $+ 1$;
13:     end if
14: end for
15: Return $\tilde{\sigma}(c) \leftarrow \frac{\text{Count}}{N}$;
```

3.4 LDS Versus MCS

In its basic form MCS method uses pseudo-random sequences. The main limitation that impacts its accuracy and scalability when estimating stability is the clumping that occurs when selecting the subsets based on the random or pseudo-random sequence. The reason for this clumping is that the different selected subsets know nothing about each other, which in turn makes them lie very close together in certain regions of the power set space while other regions have no selected subsets. As it has been shown in [22], about $\sqrt{N}$ out of N sample points could lie in clumps. This means that we need four times as many samples to halve the error, which dramatically influences the convergence rate. In fact, it has been proven that MCS method often provides a convergence rate of $O(\frac{1}{\sqrt{N}})$ using N samples [21]. Our proposed LDS method avoids the clumping by using low-discrepancy sequences instead of random or pseudo-random sequences. The former sequences often produce correlated samples in a quasi-random and deterministic fashion. Due to the correlations, those samples capture more uniformity properties and are less versatile than those samples of random or pseudo-random sequences. It has been proven in [7,21,22] that the usage of low-discrepancy sequences in sampling provides a resulting convergence rate of $O(\frac{\log N}{N})$. From a theoretical perspective, we believe that this significant convergence rate makes LDS often more accurate than MCS. For instance, after only $N = 100$ samples, MCS could have a sampling error $O(\frac{1}{\sqrt{100}}) \approx 0.1$ whereas LDS has a sampling error $O(\frac{\log 100}{100}) \approx 0.02$.

4 Experimental Evaluation

The main goal of our experimental evaluation is to investigate the following key questions: (Q1) Is LDS more accurate than MCS for estimating stability even with small formal contexts? (Q2) Is LDS scalable when approximating the stability on large-sized concepts?

4.1 Methodology

We started our experiments by first selecting three real-life datasets: (1) *Mushroom* which describes mushroom species according to a set of features [3]; (2) *Phytotherapy* which presents diseases and medicinal plants that treat them [1]; (3) *Woman-Southern-Davis*[3] which describes a set of Southern women and the events they attended) [8]. Table 1 briefly summarizes these datasets.

Table 1. A brief description of tested datasets.

Dataset	No. objs	No. attrs	Max. intent	Lattice size
Mushroom	8416	128	22	238,710
Phytotherapy	142	108	11	304
WomenDavis	18	14	8	67

To evaluate our proposed LDS algorithm, we compared the results of the following five tested algorithms:

- **Sobol**: LDS algorithm based on Sobol sequence.
- **Scrambled VDC-2**: LDS algorithm based on Scrambled Van der corput sequence with radical base 2.
- **Scrambled VDC-3**: LDS algorithm based on Scrambled Van der corput sequence with radical base 3.
- **Scrambled VDC-5**: LDS algorithm based on Scrambled Van der corput sequence with radical base 5.
- **Monte Carlo**: MCS algorithm — which can serve as a good baseline here.

To find robust answers to the proposed questions, we conducted our empirical study with the chosen algorithms under the following setting:

- Vary the number of samples N from 1 up to 100. This could be helpful to precisely judge the convergence rate.

[3] Publicly available: http://fimi.ua.ac.be/data/.

- Re-run Monte Carlo algorithm for a maximum number of 1000 iterations. This is important to properly assess Monte Carlo which is purely random and sensitive to the starting point.
- Consider the concepts with the maximum intent sizes in order to asses how LDS can perform in extreme (or large) cases.

We then assess the accuracy and scalability of results by recording both the estimated stability and the elapsed time obtained at each sample size. These results are then used to calculate the following two metrics over intent sizes:

(1) The average absolute error ξ_a of estimated stability, which is computed as:

$$\xi_a = \frac{1}{|\mathcal{C}|} \sum_{c_k \in \mathcal{C}} |\tilde{\sigma}(c_k) - \sigma(c_k)| \tag{10}$$

where $\mathcal{C}$ is the set of concepts with the same intent size a, and $|\mathcal{C}|$ is its cardinality. $\tilde{\sigma}(c_k)$ is the estimated stability of the concept c_k and $\sigma(c_k)$ is the exact value of stability.

(2) Average elapsed time τ_a, which is computed as:

$$\tau_a = \frac{1}{|\mathcal{C}|} \sum_{c_k \in \mathcal{C}} t_k \tag{11}$$

where t_k is the elapsed time for approximating the stability of concept c_k.

All the experiments were run on an Intel(R) Core(TM) i7-2600 CPU @ 3.20 GHz computer with 16 GB of memory under Windows 10. We implemented our LDS algorithm as an extension to the Concepts 0.7.11 package that is implemented by Sebastian Bank.[4] For generating low-discrepancy sequences, we used SOBOL library.[5]

4.2 Results and Discussion

In terms of accuracy and performance, the results in Fig. 1 illustrate that MCS is always less accurate than all the tested variants of LDS, including Sobol and Scrambled VDC at bases 2, 3 and 5 on large data sets such as mushroom with large-sized intents. The same conclusion applies for medium and small datasets with concepts of small-sized intents such as Phytotherapy with intent size 3 (see Fig. 2) and Woman-Southern-Davis with intent sizes 4, 5 and 7 (see Fig. 3).

In terms of convergence behavior, it is clear that Scrambled VDC-3 and Scrambled VDC-5 compete at different intent sizes, both of them achieve the most accurate convergence rate of all the algorithms compared on the three underlying datasets. Scrambled VDC-2 and Sobol have frequently a similar accuracy and stable convergence behavior on large-sized intents. This may be due

[4] Publicly available: https://pypi.python.org/pypi/concepts.
[5] Publicly available: http://people.sc.fsu.edu/~jburkardt/py_src/sobol/sobol.html.

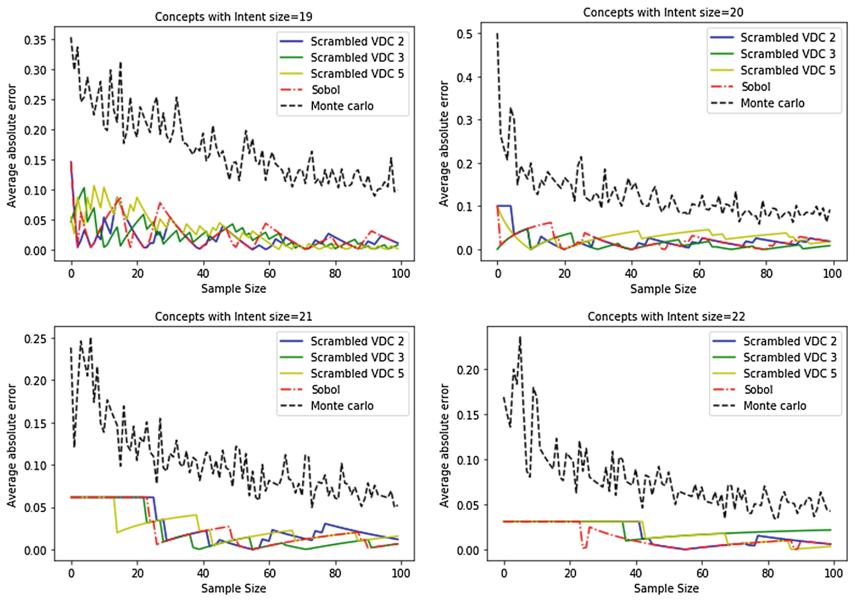

Fig. 1. The average absolute error ξ_a of LDS and MCS on *Mushroom* with intent size $a \in \{19, 20, 21, 22\}$. Each plot is labelled with the size of intent.

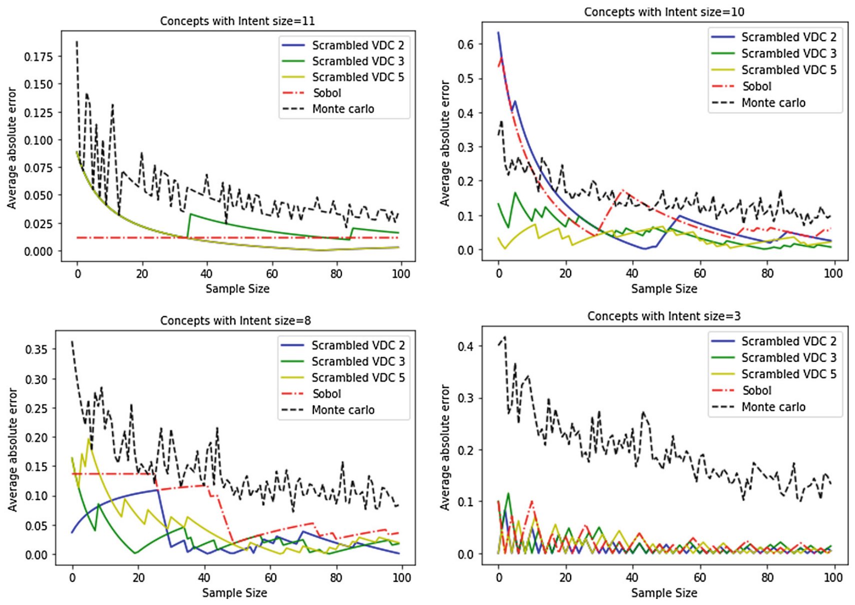

Fig. 2. The average absolute error ξ_a of LDS and MCS on *Phytotherapy* with intent size $a \in \{3, 8, 10, 11\}$. Each plot is labelled with the size of intent.

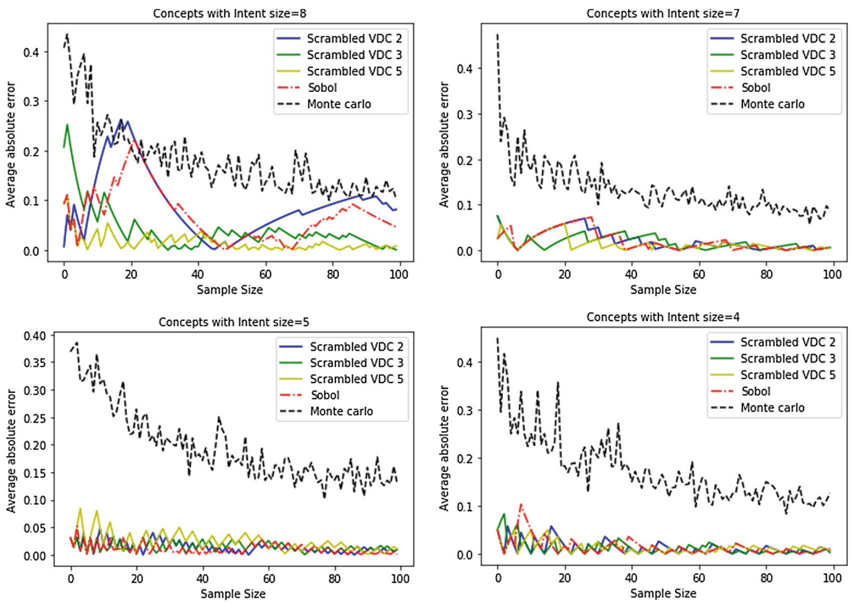

Fig. 3. The average absolute error ξ_a of LDS and MCS on *Woman-Southern-Davis* with intent size $a \in \{4, 5, 7, 8\}$. Each plot is labelled with the size of intent.

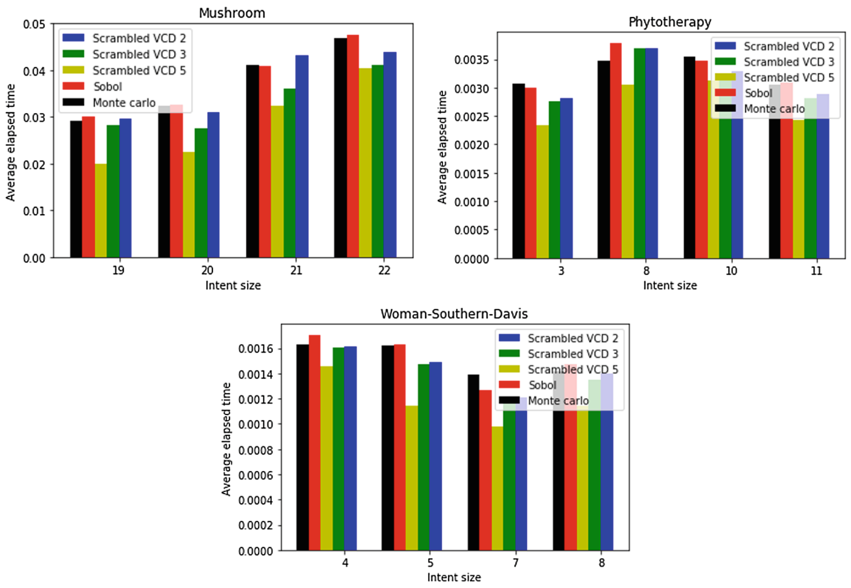

Fig. 4. The average elapsed time τ_a (in mins) for LDS and MCS on *Mushroom*, *Phytotherapy* and *Woman-Southern-Davis*.

to the fact that both of them use a binary representation (i.e., base 2) in the sequence generation process. MCS has often an unstable convergence rate when the sample size is less than 90, which could explain its need to increase the sample size for a better convergence behavior.

In terms of computational time, the results in Fig. 4 show that Scrambled VDC-5 dominates all other tested algorithms on all intent sizes. This is due to the fact that Scrambled VDC generates its sequence points using radical base 5 which requires less computational time than base 2 or 3. Apart from Scrambled VDC-5, none of the tested algorithms—the Sobol, the scrambled VDC at bases 2 and 3 and MCS — shows its clear superiority over the others.

Overall, the results in Figs. 1, 2, 3 and 4 clearly show that LDS outperforms MCS for approximating stability on all underlying datasets. Under the same number of samples, LDS can converge to more accurate estimations than MCS in the presence of large-sized concepts, and even with less computational time using some LDS variants (e.g., Scrambled VDC-5).

5 Conclusion

We have proposed LDS, a two-stage method that exploits LDS to efficiently estimate the stability index. To improve the accuracy of sampling, low-discrepancy (or quasi-random) sequences are first generated in order to eliminate the clumping of the sampled subsets and allow the uniform selection of correlated subsets across all the areas of the intent (or extent) power set space. We focus here on using Scrambled Van der Corput and Sobol sequences but the LDS method is applicable to other low-discrepancy sequences in general. Experiments on real-life datasets show that LDS is able to greatly bound the convergence rate by several orders of magnitude compared to MCS, which is the commonly used method in FCA to approximate stability. In the future we plan to integrate LDS with other variance reduction techniques such as importance sampling and stratification sampling. We also intend to perform an on-line sampling scenario of LDS-based algorithm to tackle stream formal contexts.

Acknowledgment. We acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1. Alpen, É.: Précis de Phytotérapie. Édition Alpen (2010). www.alpen.mc/precis-de-phytotherapie
2. Babin, M.A., Kuznetsov, S.O.: Approximating concept stability. In: Domenach, F., Ignatov, D.I., Poelmans, J. (eds.) ICFCA 2012. LNCS (LNAI), vol. 7278, pp. 7–15. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29892-9_7
3. Bache, K., Lichman, M.: Mushroom data set (2013). http://archive.ics.uci.edu/ml
4. Belohlavek, R., Macko, J.: Selecting important concepts using weights. In: Valtchev, P., Jäschke, R. (eds.) ICFCA 2011. LNCS (LNAI), vol. 6628, pp. 65–80. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20514-9_7

5. Buzmakov, A., Kuznetsov, S.O., Napoli, A.: Is concept stability a measure for pattern selection? Procedia Comput. Sci. **31**, 918–927 (2014)
6. Buzmakov, A., Kuznetsov, S.O., Napoli, A.: Scalable estimates of concept stability. In: Glodeanu, C.V., Kaytoue, M., Sacarea, C. (eds.) ICFCA 2014. LNCS (LNAI), vol. 8478, pp. 157–172. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07248-7_12
7. Caflisch, R.E.: Monte Carlo and quasi-Monte Carlo methods. Acta Numer. **7**, 1–49 (1998)
8. Davis, A., Gardner, B., Gardner, M.: Deep South (1941). http://networkdata.ics.uci.edu/netdata/html/davis.html
9. Faure, H., Tezuka, S.: Another random scrambling of digital (t, s)-sequences. In: Fang, K.T., Niederreiter, H., Hickernell, F.J. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2000, pp. 242–256. Springer, Heidelberg (2002). https://doi.org/10.1007/978-3-642-56046-0_16
10. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, New York (1999). https://doi.org/10.1007/978-3-642-59830-2. Translator C. Franzke
11. Klimushkin, M., Obiedkov, S., Roth, C.: Approaches to the selection of relevant concepts in the case of noisy data. In: Kwuida, L., Sertkaya, B. (eds.) ICFCA 2010. LNCS (LNAI), vol. 5986, pp. 255–266. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11928-6_18
12. Kuipers, L., Niederreiter, H.: Uniform distribution of sequences. Courier Corporation (2012)
13. Kuznetsov, S., Obiedkov, S., Roth, C.: Reducing the representation complexity of lattice-based taxonomies. In: Priss, U., Polovina, S., Hill, R. (eds.) ICCS-ConceptStruct 2007. LNCS (LNAI), vol. 4604, pp. 241–254. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73681-3_18
14. Kuznetsov, S.O.: Learning of simple conceptual graphs from positive and negative examples. In: Żytkow, J.M., Rauch, J. (eds.) PKDD 1999. LNCS (LNAI), vol. 1704, pp. 384–391. Springer, Heidelberg (1999). https://doi.org/10.1007/978-3-540-48247-5_47
15. Kuznetsov, S.O.: On stability of a formal concept. Ann. Math. Artif. Intell. **49**(1), 101–115 (2007)
16. Kuznetsov, S.O., Makhalova, T.P.: Concept interestingness measures: a comparative study. In: Proceedings of the Twelfth International Conference on Concept Lattices and Their Applications, Clermont-Ferrand, France, 13–16 October 2015, pp. 59–72 (2015)
17. Kuznetsov, S.O., Makhalova, T.P.: On interestingness measures of formal concepts. CoRR abs/1611.02646 (2016)
18. Landau, D.P., Binder, K.: A Guide to Monte Carlo Simulations in Statistical Physics. Cambridge University Press, Cambridge (2014)
19. Lemieux, C.: Monte Carlo and quasi-Monte Carlo sampling (2009)
20. Muangprathub, J.: A novel algorithm for building concept lattice. Appl. Math. Sci. **8**(11), 507–515 (2014)
21. Niederreiter, H.: Random Number Generation and Quasi-Monte Carlo Methods. SIAM, Philadelphia (1992)
22. Owen, A.B.: Monte Carlo extension of quasi-Monte Carlo. In: Simulation Conference Proceedings Winter, vol. 1, pp. 571–577. IEEE (1998)
23. Roth, C., Obiedkov, S., Kourie, D.G.: On succinct representation of knowledge community taxonomies with formal concept analysis. Int. J. Found. Comput. Sci. **19**(02), 383–404 (2008)

24. Schretter, C., He, Z., Gerber, M., Chopin, N., Niederreiter, H.: Van der corput and golden ratio sequences along the hilbert space-filling curve. In: Cools, R., Nuyens, D. (eds.) Monte Carlo and Quasi-Monte Carlo Methods, vol. 163, pp. 531–544. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33507-0_28
25. Zhi, H.L.: On the calculation of formal concept stability. J. Appl. Math. **2014**, 1–6 (2014)