Scalable Sampling for High Utility Patterns

Lamine Diop*, Marc Plantevit*
*EPITA Research Laboratory (LRE), Le Kremlin-Bicetre, Paris FR-94276, France, firstname.lastname@epita.fr

***Abstract*—Discovering valuable insights from data through meaningful associations is a crucial task. However, it becomes challenging when trying to identify representative patterns in quantitative databases, especially with large datasets, as enumeration-based strategies struggle due to the vast search space involved. To tackle this challenge, output space sampling methods have emerged as a promising solution thanks to its ability to discover valuable patterns with reduced computational overhead. However, existing sampling methods often encounter limitations when dealing with large quantitative database, resulting in scalability-related challenges. In this work, we propose a novel high utility pattern sampling algorithm and its on-disk version both designed for large quantitative databases based on two original theorems. Our approach ensures both the interactivity required for user-centered methods and strong statistical guarantees through random sampling. To demonstrate the interest of our approach, we present a compelling use case involving archaeological knowledge graph sub-profiles discovery. Experiments on semantic and none-semantic quantitative databases show that our approach outperforms the state-of-the art methods.**

***Index Terms*—Knowledge discovery, Output pattern sampling, High utility itemset**

I. Introduction

Exploratory Data Analysis (EDA) faces growing challenges due to the increasing complexity and scale of datasets. Recently, methods for constructing and exploring database profiles for knowledge graphs have been developed to support interactive mining and visualization [1], [2]. These profiles capture crucial information by assigning weights to predicates linking two classes, making the graph more interpretable and user-friendly. Presented as a network, the profile becomes easier to understand and interact with, offering zoom, pan, and filter capabilities. However, visualizing profiles derived from large knowledge graphs remains difficult with existing online tools due to the complexity of interactivity, which allows users to explore patterns at various levels of granularity, incorporating predicate utilities and external weights. In EDA, the discovery of useful weighted patterns in databases with weighted items—known as High Utility Pattern (HUP) and high average-utility (HAUP) mining [3], [4]—enhances data interpretability. Visualization techniques, like networks, effectively convey the essence of these patterns, as shown in Figure 1, but they require fast, iterative exchanges of information between the system and the user to be effective [5].

To overcome these obstacles, researchers have proposed new approaches such as sampling-based approximate mining methods [6] and output sampling methods [7], [8], [9]. These innovative techniques enable the extraction of meaningful patterns based on a probability distribution, ensuring both representativeness and control over the output space. Despite their widespread application in frequent pattern mining tasks, they have seen limited utilization within the domain of qDB [10], [11]. This discrepancy can be attributed to the intricate and memory-intensive nature of the weighting phase involved in handling transactions within qDB. Consequently, the complexity of these methods hinders their scalability for large databases. This limitation becomes particularly problematic when exploring very large databases and motivate this paper.

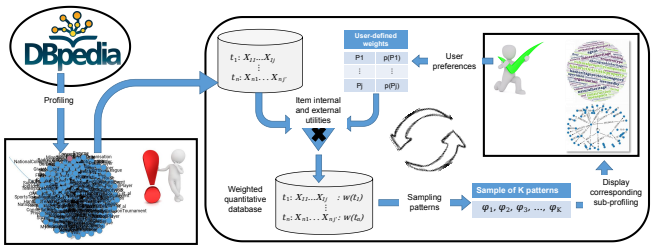

Fig. 1: Discovery of sub-profiles in knowledge graphs

Our work introduces QPLUS, a high-performance algorithm for exact sampling of high utility patterns (HUP/HAUP) with optional length constraints, complemented by an on-disk version, QPLUSDISK, for handling large databases efficiently. We propose two original theorems alongside the Upper Triangle Utility (UTU) concept, which allows efficient utility pattern computation without materializing storage, facilitating large qDB exploration. Additionally, we demonstrate the application of our methods in weighted knowledge graphs, using sampled high utility patterns to extract representative sub-profiles through transformation into quantitative databases.

The outline of this paper is as follows. Section II reviews related works about HUP/HAUP mining approaches and pattern sampling methods. Section III introduces basic definitions and the formal problem statement. We present in Section IV our contribution for pattern sampling in qDB. We evaluate our approach in Section V and conclude in Section VI.

II. Related Work

This section presents the HUP/HAUP mining in qDBs literature and output space pattern sampling.

A. HUP and HAUP mining

Mining HUP/HAUP from qDB faces challenges such as candidate pattern restriction and high computational cost, with various methods proposed to address these issues [12]. However, scalability remains hindered for large-scale, diverse databases, and the long-tail problem complicates exact HUP

mining. Approaches like average utility measures [13] aim to mitigate this but may introduce bias. Despite these challenges, HUP mining is vital for extracting insights from qDBs, with novel sampling techniques [10], [11] offering solutions to enhance efficiency and decision-making.

B. Output space HUP/HAUP sampling

Recently, the HAISAMPLER algorithm [10] extends output pattern sampling to qDB. Despite its success, scalability in large databases is hindered by intensive weight calculations, making it computationally demanding and memory-intensive. More recently, HUPSAMPLER [11], a hybrid approach, extracts High Utility Patterns (HUP) from qDBs, introducing interval constraints and a random tree-based pattern growth technique. While similar to HAISAMPLER [10], HUPSAMPLER uniform pattern drawing during sampling can compromise representativeness while the phase of building a tree remains time consuming with large databases.

We propose a novel pattern sampling method for large qDBs that ensures efficient, exact sampling with or without length constraints, drawing patterns based on utility without intensive item weighting, making it scalable and memory-efficient.

III. PROBLEM STATEMENT

This section introduces fundamental concepts and notations, providing the necessary definitions to help readers.

A. Basic definitions and notations

A knowledge graph is a semantic graph combining a TBox (Terminological Box) and an ABox (Assertional Box). The TBox defines the schema, including classes, properties, and constraints. The ABox contains the actual data, comprising instances of classes and the relationships between them.

a) From knowledge graph profile to qDB: Given a knowledge graph $\mathcal{K}$ built upon a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$, i.e., $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, a profile is defined as a set of pairs $((\mathcal{S}, P, \mathcal{O}), w)$ such that: $\mathcal{S}$ is a set of subjects that share a same number of w instances of the set of objects $\mathcal{O}$ via the predicate P. In other words, a profile encodes the types (elements of $\mathcal{T}$) and the terms (instances of thesauri) of a knowledge graph into a compact structure while aggregating their statistics like the number of triples.

Example 1. *Let P_1="r:represent", P_2="r:encompass", P_3="r:involve" be predicates, C_1="r:Encounter_Event", C_2="r:Man-Made", C_3="r:Document", C_4="r:Site" concepts, and e_1="anastylosis", e_2="geoarchaeology", e_3="excavation" terms. We build in Figure 2 a toy profile.*

MCMC-based methods [7] have been used to address output sampling in unweighted graph data, but they often suffer from time-consuming convergence. An alternative, Random Sampling, lacks guarantees on representativeness, as our baseline BOOTSTRAP. To overcome these problems, our transformation approach effectively captures the semantic relations within the entire knowledge graph.

To apply our approach, we need first to convert the knowledge graph profile into a qDB $\mathcal{D}$. Therefore, we consider the

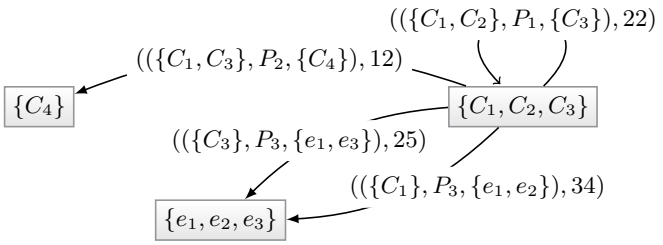

Fig. 2: Toy knowledge graph profile

weighted items of each quantitative transaction as the set of pairs $((\mathcal{S}, P, \mathcal{O}), w)$ having a same node source (or target) in $\mathcal{K}$. Let $\mathcal{I} = \{X_1, \cdots, X_N\}$ be a finite set of items with an arbitrary total order $\prec$ between them: $X_1 \prec \cdots \prec X_N$. A pattern, denoted as $\varphi = X_{j_1} \cdots X_{j_n}$, where $n \leq N$, refers to a non-empty subset of $\mathcal{I}$, i.e., $\varphi \subseteq \mathcal{I}$. The pattern language corresponds to $\mathcal{L} = 2^{\mathcal{I}} \setminus \emptyset$, and the length of a pattern $\varphi = X_{j_1} \cdots X_{j_n}$, $\varphi \in \mathcal{L}$, denoted as $|\varphi| = n$.

Definition 1 (Semantic quantitative transaction and qDB)**.** *A semantic quantitative transaction is a set of weighted items $t = \{X_{j_1} : \omega_{j_1} \cdots X_{j_n} : \omega_{j_n}\}$ sharing either the same node source or the same node target in the knowledge graph profile, with ω_{j_k} the quantity of X_{j_k} in t denoted by $q(X, t)$. A qDB $\mathcal{D}$ corresponds to a multi-set of quantitative transactions.*

We denote $t^i = X_{j_1} \cdots X_{j_i}$ as an itemset formed by the i first items of t. Thus, we have $|t^i| = i$. The i^{th} item of the transaction t is denoted by $t[i]$.

Example 2. *Here are two quantitative transactions that can be generated from the toy profile in Figure 2: $t_1 = \{X_1 : 22, X_2 : 12, X_3 : 25, X_4 : 34\}$ with outgoing predicates, and $t_2 = \{X_1 : 22\}$ with incoming predicates, with $X_1 = (\{C_1, C_2\}, P_1, \{C_3\})$, $X_2 = (\{C_1, C_3\}, P_2, \{C_4\})$, $X_3 = (\{C_3\}, P_3, \{e_1, e_3\})$, $X_4 = (\{C_1\}, P_3, \{e_1, e_2\})$. The remaining transactions are $t_3 = \{X_2{:}12\}$ and $t_4 = \{X_3{:}25, X_4{:}34\}$, then our toy qDB is $\mathcal{D} = \{t_1, t_2, t_3, t_4\}$.*

b) User-defined Weights: In refining subprofile discovery through pattern sampling, we incorporate user-defined strict positive weights for each predicate P of a given item X denoted by $p(X)$, allowing users to tailor and bias the subprofile view based on individual priorities. This addition provides flexibility to our profile visualization tool, enabling users to customize displayed patterns for a personalized and adaptable exploration experience. Let us consider, in our example, that $p(X_1) = 2$, $p(X_2) = 1$, and $p(X_3) = p(X_4) = 3$.

Definition 2 (Pattern utility in a transaction)**.** *Given a transaction t of a qDB $\mathcal{D}$, the weight of the item X in t, denoted as $\omega(X, t)$, is given by: $\omega(X, t) = q(X, t) \times p(X)$. The utility of a pattern φ of $\mathcal{L}(\mathcal{D})$ in t is defined as follows:*

$u_{\mathcal{D}}(\varphi, t) = \sum_{X \in \varphi} \omega(X, t)$ *if $\varphi \subseteq t$ and 0 otherwise.*

c) From utility patterns to knowledge graph sub-profile: After that, each sampled pattern is converted into a sub-

profile graph where the nodes sharing at least one type or term are merged to form a maximal profile as defined in [2]. For instance, if the pattern $\varphi = X_2X_4$ is drawn, then it corresponds to the knowledge graph sub-profile in Figure 3.

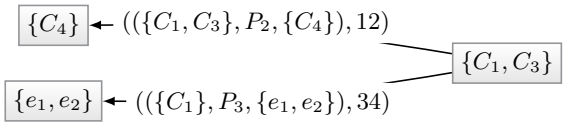

Fig. 3: Sub-profile from the pattern X_2X_4

Based on the previous basic notions, we can formalize the problem of output space pattern sampling in qDBs.

B. HUP/HAUP sampling problem

By definition, a pattern sampling method aims to randomly select a pattern φ from a language $\mathcal{L}$ based on an interestingness measure m. The notation $\varphi \sim \mathbb{P}(\mathcal{L})$ represents the selection such a pattern, where $\mathbb{P}(\cdot) = m(\cdot)/Z$ is a probability distribution over $\mathcal{L}$, and Z is the normalization constant. In our case, we specifically focus on high utility as an intuitive measure of interestingness, which allows experts to capture the most representative subsets in the qDB.

Definition 3 (Utility of a pattern)**.** *Given a qDB $\mathcal{D}$ and a pattern φ defined in $\mathcal{L}(\mathcal{D})$, the utility of φ in $\mathcal{D}$, denoted as $\mathbb{U}_{\mathcal{L}}(\varphi, \mathcal{D})$, is defined as follows:* $\mathbb{U}_{\mathcal{L}}(\varphi, \mathcal{D}) = \sum_{t \in \mathcal{D}} u_{\mathcal{D}}(\varphi, t)$.

For generality, we additionally consider utilities that are independent of any specific database, referred to as the length-based utility [10]. We use $\mathbb{L}_{[\mu..M]}(\cdot) : \mathbb{N}^* \to \mathbb{R}^+$, where μ and M are two positive integers with $\mu \leq M$, to define the length-based utility function. In any case, when this function is used, $\mathbb{L}_{[\mu..M]}(\ell) = 0$ if $\ell \notin [\mu..M]$. Consequently, if $\mathbb{L}_{[\mu..M]}(\ell) = \frac{1}{\ell}$, then we are dealing with the sub-problem of HAUP under length constraint [10], where the utility of a pattern φ is defined as its average utility $\frac{\mathbb{U}_{\mathcal{L}}(\varphi,\mathcal{D})}{|\varphi|}$ if $|\varphi| \in [\mu..M]$, and 0 otherwise. With HUP sampling, we set the length-based utility function to 1 for any pattern in $\mathcal{L}(\mathcal{D})$. The maximum length constraint can be set to infinity ($+\infty$) to avoid length constraints. Let $\mathcal{L}_{[\mu..M]}(\mathcal{D})$ denote the set of all patterns in $\mathcal{L}(\mathcal{D})$ with a length greater than μ and lower than M, i.e., $\mathcal{L}_{[\mu..M]}(\mathcal{D}) = \{\varphi \in \mathcal{L}(\mathcal{D}) : |\varphi| \in [\mu..M]\}$.

Finally, given a qDB $\mathcal{D}$, two positive integers μ and M, our objective is to draw a pattern $\varphi \in \mathcal{L}_{[\mu..M]}$ with a probability proportional to its weighted utility in the database $\mathcal{D}$, i.e.,

$$\mathbb{P}(\varphi) = \frac{\mathbb{U}_{\mathcal{L}}(\varphi, \mathcal{D}) \times \mathbb{L}_{[\mu..M]}(|\varphi|)}{\sum_{\varphi' \in \mathcal{L}_{[\mu..M]}} \mathbb{U}_{\mathcal{L}}(\varphi', \mathcal{D}) \times \mathbb{L}_{[\mu..M]}(|\varphi'|)}.$$

IV. Generic utility pattern sampling

We adopt in this paper the multi-step pattern sampling [8] technique. Therefore, we first need to perform a preprocessing phase, which involves weighting each transaction.

A. Weighting of a qDB

Before introducing our weighting approach, let us first briefly explain our baseline weighting approach.

a) HAISampler *[10] weighting approach:* The transaction weighting phase is crucial in multi-step pattern sampling, as it computes the sum of the utility of all patterns in a transaction. HAISampler uses a weighting matrix to calculate the weight of patterns of a specific length, denoted ℓ, by aggregating the weights of patterns of length ℓ that appear in the transaction. This weighting is based on an arbitrary total order relation $\prec$. For each transaction t, every item $t[i]$ has an associated local matrix of 2 rows and L columns, where L is the maximum length of patterns that can be generated with the first i^{th} items (t^i) under the given order and length constraint. The first row in the local matrix holds the weight of patterns containing item $t[i]$, while the second row contains those patterns that do not include $t[i]$. For example, in Table I, for transaction $t_1^3 = \{X_1 : 22,\ X_2 : 12,\ X_3 : 25\}$, the sum of utilities of patterns of length 2 that contain item $t_1[3] = X_3$ is 206, whereas for patterns that do not contain X_3, it is 56. The sum of utilities of patterns of length 3 that include X_3 is 131, with no patterns of length 3 that exclude it. The total weight of the transaction is computed as $(34 + 131) + (233 + 262) + (364 + 131) + (165 + 0) = 1320$.

TABLE I: Weighting matrix of t_1 by HAISampler [10]

$t_1 : \{$

X_1
44
0

,

X_2	
12	56
44	0

,

X_3		
75	206	131
56	56	0

,

X_4			
34	233	364	165
131	262	131	0

$\}$

However, in large databases with limited available memory, the storage cost can become prohibitive. Therefore, we propose a new, efficient, and original weighting approach for qDB.

b) Our weighting approach: The solution in this paper removes the need for a local matrix for each item. We aim to create a formula to compute the weight of any transaction in a qDB based on the sum of its item utilities. To achieve this, we propose the Upper Triangle Utility weighting approach, which uses an upper triangle matrix. This matrix contains the weights of each item and allows for the efficient computation of the transaction weight and exact pattern drawing. At the intersection of line ℓ and column i, the matrix provides the sum of utilities for the patterns of length ℓ in the transaction portion t^i.

Definition 4 (Upper Triangle Utility)**.** *Given a quantitative transaction t, ℓ and i two integers, with $0 < \ell, i \leq |t|$. The upper triangle utility of t denoted by $\mathcal{V}_t$ is defined as follows:*

$\mathcal{V}_t(\ell, i) = 0$ *if* $\ell > i$, $\quad \mathcal{V}_t(1, i) = \sum_{j=1}^{i} \omega(t[j], t)$

$\mathcal{V}_t(\ell, i) = \binom{i-1}{\ell-1} \times \omega(t[i], t) + \mathcal{V}_t(\ell - 1, i - 1) + \mathcal{V}_t(\ell, i - 1)$.

The recursive formula can be interpreted easily as follows: on one hand, $\binom{i-1}{\ell-1} \times \omega(t[i], t) + \mathcal{V}_t(\ell-1, i-1)$ gives the aggregate weights of the set of patterns of length ℓ appearing in t and containing the item $t[i]$. On the other hand, $\mathcal{V}_t(\ell, i-1)$ gives that of the set of patterns of length ℓ in t without the item $t[i]$.

Example 3. *Let us compute the upper triangle utility of t_1 in Table II with Definition 4. The weight of the corresponding transaction t is the sum of the values in column $|t|$ of $\mathcal{V}_t$. Then*

TABLE II: Upper triangle utility of transaction t_1

$t_1:\{$	X_1	,	X_2	,	X_3	,	X_4	$\}$
	44		56		131		165	
	0		56		262		495	
	0		0		131		495	
	0		0		0		165	

we have $165 + 495 + 495 + 165 = 1320$. *Hence we get the same result as* HAISAMPLER.

In reality, the upper triangle utility hides wonderful properties. For example, we have the intuition that $\mathcal{V}_t(\ell, i) = \mathcal{V}_t(i - \ell + 1, i)$. Furthermore, for any couple of values (ℓ, i) such that $\ell \leq i \leq |t|$, we get the result $\frac{\mathcal{V}_t(\ell,i)}{\mathcal{V}_t(1,i)} = \binom{i-1}{\ell-1}$. Hence, we state Theorem 1.

Theorem 1. *Let $\mathcal{V}_t$ be the upper triangle utility of any given a quantitative transaction t. For any positive integers ℓ and i such that $\ell \leq i \leq |t|$, the following statement holds: $\mathcal{V}_t(\ell, i) = \binom{i-1}{\ell-1} \times \mathcal{V}_t(1, i)$.*

In addition, we can also state the following property.

Property 1 (Symmetry)**.** *If $\mathcal{V}_t$ is the upper triangle utility of a given transaction t, and ℓ and i two positive integers such that $\ell \leq i \leq |t|$ then we have: $\mathcal{V}_t(\ell, i) = \mathcal{V}_t(i - \ell + 1, i)$.*

Property 1 is particularly useful in the preprocessing and the drawing phases, especially when the maximal length constraint is either not used or set higher than half of the transaction length. Therefore, it is used automatically in our approach. In the case where there is no maximal length constraint (i.e., $M = \infty$) while the minimal one is set to $\mu = 1$, Theorem 2 can be used to compute the weigh of each transaction.

Theorem 2. *Let t be a transaction from a qDB. The cumulative utility sum of the entire set of patterns that appear in transaction t is expressed as: $W(t) = 2^{|t|-1} \times \sum_{X \in t} \omega(X, t)$.*

Based on these theorems, we no longer need to store the UTU in memory. This is because we can easily deduce any value $\mathcal{V}_t(\ell, i)$ based on $\mathcal{V}_t(1, i) = \sum_{j=1}^{i} \omega(t[j], t)$.

B. Drawing a high utility pattern

Sampling a pattern can be done efficiently from the upper triangle utility. Let us first recall the definition of the probability to select an item at a given position.

Definition 5. *Let t be a quantitative transaction, φ be the already drawn k items ordered according to $\prec$, and ℓ be the rest of the items to select, $\varphi = X_{j_k} \cdots X_{j_1}$. The probability to pick the item $t[i]$, with $i < j_k$, given φ and ℓ, is defined as the ratio of the aggregate weights of all patterns of length $\ell + k$ in the transaction $t^i \cup \varphi$, containing the pattern $t[i] \cup \varphi$, equals to $\mathcal{V}_t(\ell, i) - \mathcal{V}_t(\ell, i-1) + \binom{i-1}{\ell-1} \times \mathbb{U}_{\mathcal{L}}(\varphi, t)$, and the total aggregate weights of all patterns of length $\ell + k$ appearing in the transaction $t^i \cup \varphi$, containing the pattern φ, equals to $\mathcal{V}_t(\ell, i) + \binom{i}{\ell} \times \mathbb{U}_{\mathcal{L}}(\varphi, t)$. Formally, we have:*

$$\mathbb{P}(t[i] | \varphi \wedge \ell) = \frac{\mathcal{V}_t(\ell, i) - \mathcal{V}_t(\ell, i-1) + \binom{i-1}{\ell-1} \times \mathbb{U}_{\mathcal{L}}(\varphi, t)}{\mathcal{V}_t(\ell, i) + \binom{i}{\ell} \times \mathbb{U}_{\mathcal{L}}(\varphi, t)}.$$

This means that a pattern φ is drawn proportionally to its utility in a given transaction t.

However, we can see that this approach needs to generate a random variable at each visited item, and it is possible to visit all items of the transaction during the drawing of a pattern. For instance, if in t_5 the drawn pattern contains the first item $t_5[1] = A$, we need a more efficient approach that does not require generating more than $|\varphi|$ random variables. Therefore, we propose an approach based on dichotomous random search.

Property 2 (Dichotomous random search)**.** *Let t be a quantitative transaction, and $t[j]$ be the last picked item during the drawing process. Using a sequential random variable generation to draw the next item $t[i]$, with $i < j$, to add in the sampled pattern φ is equivalent to jumping directly on it based on dichotomous search. In other words, if a random number α_i drawn from the interval $]0, \mathcal{V}_t(\ell, i) + \binom{i}{\ell} \times \mathbb{U}_{\mathcal{L}}(\varphi, t)]$ allows us to pick $t[i]$, i.e., $\frac{\alpha_i}{\mathcal{V}_t(\ell,i) + \binom{i}{\ell} \times \mathbb{U}_{\mathcal{L}}(\varphi,t)} \leq \mathbb{P}(t[i] | \varphi \wedge \ell)$, then from position j of t, we can directly select the item $t[i]$ such that $\mathcal{V}_t(\ell, i-1) + \binom{i-1}{\ell} \times \mathbb{U}_{\mathcal{L}}(\varphi, t) < \alpha_i \leq \mathcal{V}_t(\ell, i) + \binom{i-1}{\ell-1} \times \mathbb{U}_{\mathcal{L}}(\varphi, t)$.*

Property 2 significantly improves efficiency for long transactions by enabling direct jumps to relevant positions, unlike the HAISAMPLER iterative approach, which can be time-consuming as it visits all items in the transaction.

C. Overview of the algorithm

QPLUS takes a qDB $\mathcal{D}$ with a total order relation $\prec$, along with positive integers μ and M. The goal is to output a pattern φ drawn proportionally to its utility, which is weighted by a length-based utility measure. It consists of two phases: a preprocessing phase and a drawing phase. In the preprocessing phase, weights are computed for each transaction based on utility and length functions, and in the drawing phase, a transaction and pattern are drawn proportionally to their weights, with items selected iteratively based on utility and length until the pattern is formed.

D. Theoretical analysis of the method

In this section, we first demonstrate the soundness of QPLUS before evaluating its time complexity (preprocessing and the sampling). Finally, we give its storage complexity.

Property 3 (Soundness)**.** *Let $\mathcal{D}$ be a qDB with an arbitrary total order relation $\prec$, and μ and M be two positive integers such that $\mu \leq M$. Algorithm* QPLUS *directly draws a pattern from $\mathcal{D}$ with a length in the range $[\mu..M]$ according to a distribution that is proportional to its weighted utility.*

The preprocessing complexity of QPlus is $O(|\mathcal{D}| \times M)$ when a maximum length is specified, and $O(|\mathcal{D}|)$ if $M = \infty$, offering a more efficient approach than HAISAMPLER. Drawing

TABLE III: Statistics of none-semantic qDB benchmark

Database	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{I}\rvert$	$\lvert t\rvert_{\min}$	$\lvert t\rvert_{\max}$	$\lvert t\rvert_{avg}$	$\mathcal{D}$ size in memory	$\mathcal{D}$ size on disk	C_n^k in memory (Prop. ??, M =10)
Retail	88,162	16,470	1	76	10.30	242.80 MB	7.36 MB	30.93 MB
BMS2	77,512	3,340	1	161	1.62	99.07 MB	3.55 MB	69.82 MB
Kosarak	990,002	41,270	1	2,497	8.09	2.13 GB	61.39 MB	1.17 GB
ChainstoreFIM	1,112,949	46,086	1	170	7.23	2.12 GB	74.48 MB	73.87 MB
USCensus	1,000,000	316	48	48	48.00	$\approx$ 10 GB	352.60 MB	18.05 MB
t17Mi20Md	20,000,000	16,957,575	14	94	52.77	Out of memory	7.83 GB	39.12 MB

Algorithm 1 QPLUS: Quantitative Pattern Sampling

Input: A qDB $\mathcal{D}$ with an arbitrary total order relation $\prec$, and two positive integers μ and M such that $\mu \leq M$
Output: A pattern $\varphi \sim \mathbb{U}_{\mathcal{L}}(\mathcal{L}_{[\mu..M]}(\mathcal{D}))$
//Preprocessing
$W_{[\mu..M]}(t) = \sum_{\ell=\mu}^{M}(\mathcal{V}_t(\ell, |t|) \times \mathbb{L}_{[\mu..M]}(\ell))$ for $t \in \mathcal{D}$
//Drawing
Draw a transaction $t : t \sim W_{[\mu..M]}(\mathcal{D}) = \sum_{t\in\mathcal{D}} W_{[\mu..M]}(t)$
Draw an integer ℓ from μ to M with $\mathbb{P}(\ell|t) = \frac{\mathcal{V}_t(\ell,|t|)\times\mathbb{L}_{[\mu..M]}(\ell)}{W_{[\mu..M]}(t)}$
$\varphi \leftarrow \emptyset$, $j \leftarrow |t|$
while $\ell > 0$ **do** ▷ Process of drawing a pattern of length ℓ
  $\alpha \leftarrow \text{random}(0, 1) \times (\mathcal{V}_t(\ell, j) + \binom{j-1}{\ell-1} \times \mathbb{U}_{\mathcal{L}}(\varphi, t))$
  $i \leftarrow \arg_i(b_{inf}(i) < \alpha \leq b_{sup}(i))$
  with $b_{inf}(i) = \mathcal{V}_t(\ell, i-1) + \binom{i-1}{\ell} \times \mathbb{U}_{\mathcal{L}}(\varphi, t)$
  and $b_{sup}(i) = \mathcal{V}_t(\ell, i) + \binom{i-1}{\ell-1} \times \mathbb{U}_{\mathcal{L}}(\varphi, t)$
  $\varphi \leftarrow \{t[i]\} \cup \varphi$ and $\ell \leftarrow \ell - 1$
  $j \leftarrow i$ ▷ Jumping to the position indexed by i
return φ ▷ Drawn proportionally to its weighted utility

complexity for QPlus is $O(K \times (\log(|\mathcal{D}|) + M \times \log(|\mathcal{I}| - M + 1)))$ for K patterns, which outperforms HAISAMPLER $O(K \times (\log(|\mathcal{D}|) + |\mathcal{I}|))$. Storage complexity is dominated by $O(||\mathcal{D}||)$ for the database itself, with additional storage $O(|\mathcal{D}| + G(|t|_{\max}))$ for transaction weights and combination values, significantly smaller than the total database storage.

We recall that our algorithm QPLUS depends on the available memory to perform the dedicated task, which is a problem when we need to explore larger qDBs. To overcome this issue, we propose an on disk approach, denoted by QPLUSDISK (see extended version for details).

V. EXPERIMENTAL STUDY

Table III outlines the characteristics of the qDBs used in our study, sourced from SPMF[1] and IBMGenerator[2]. The framework[3], implemented in Python 3.0 on Google Colab with 12 GB RAM, evaluates execution times and utility for patterns with length constraints between [2..10] and without constraints for comparison with HAISAMPLER. We use an average utility function for evaluation, repeating execution 10 times with negligible standard deviation.

A. Analysis of our approaches under length constraint

QPLUS outperforms HAISAMPLER in preprocessing speed (15-45 times faster) and handles large datasets efficiently, processing 2 million transactions in 5 minutes. For pattern drawing, QPLUS is 4 to 16 times faster, particularly for long transactions, and excels in sampling due to its UTU-based acceptance probability, maintaining superior speed even with large datasets.

B. HUP-based sub-profiles in knowledge graph

Our approach enables the discovery of diverse, representative sub-profiles within knowledge graph profiles, offering flexibility in sub-profile size ($M \times K$) for targeted visualization and clearer insights, particularly for complex profiles like DOREMUS, BENICULTURALI, and DBpedia.

TABLE IV: Semantic qDB of Knowledge graph profiles

$\mathcal{A}$	statistics for TT profile		
	nb. of TT	nb. of links	nb. of nodes
DOREMUS	2,399	1,785	115
BENICULTURALI	641	7,518	440
DBpedia	404	6,010	204

a) Qualitative evaluation of sampled pattern-based sub-profile: Figure 5 shows that QPLUS outperforms BOOTSTRAP by producing higher average utility and a richer variety of distinct patterns ($M = 5$), demonstrating its effectiveness in generating representative sub-profiles for large knowledge graph profiles. Figure 6 shows that QPLUS offers more stable and reliable pattern utility estimates compared to BOOTSTRAP, which exhibits higher variance and less precision, highlighting QPLUS advantage in generating consistent sub-profile samples. Figure 7 visualizes sub-profiles from QPLUS-sampled patterns, showing dense ego-network structures in DOREMUS and DBpedia around nodes like 'prov#Entity' and 'Company', while BENICULTURALI forms a more distributed network with no isolated nodes.

C. Response time for interactive sub-profile discovery

We integrate QPLUS into a user-centric sub-profile discovery framework, achieving fast processing (e.g., 0.004 to 0.023 seconds) and drawing times (e.g., 0.069 to 0.141 milliseconds) for large qDBs, enabling sub-profile generation with hundreds of nodes in under 1 millisecond, crucial for interactive user experiences with large databases.

VI. CONCLUSION

This paper introduces a groundbreaking approach—the first of its kind—for extracting sub-profiles from knowledge graph profiles, employing high utility patterns and an output sampling method. With the presentation of two original theorems, we achieve an efficient computation of the total sum of utility patterns for a given quantitative transaction. The concept of Upper Triangle Utility (UTU), embodied by a non-materialized upper triangle matrix, emerges as a pivotal

[1]https://www.philippe-fournier-viger.com/spmf/index.php
[2]https://github.com/zakimjz/IBMGenerator
[3]https://github.com/ScalableSamplingInLargeDatabases/QPlus

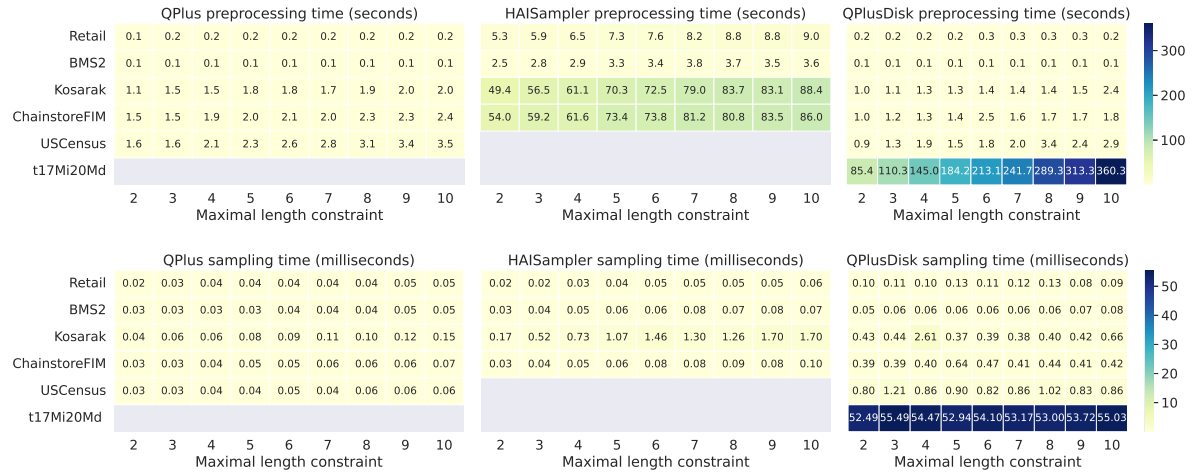

Fig. 4: Comparing the evolution of execution time based on maximum length constraint (in gray = out of memory)

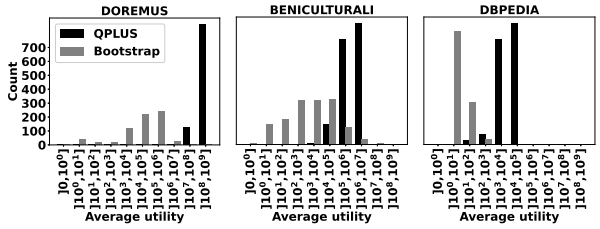

Fig. 5: Representativeness of 1,000 drawn patterns by QPLUS and Bootstrap

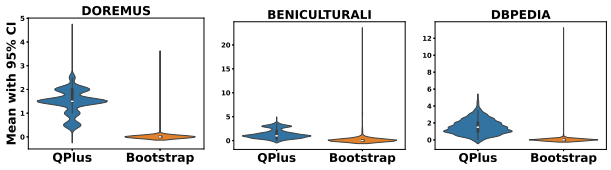

Fig. 6: Violin plots with mean and 95% confidence intervals for the plus-value of QPLUS on Bootstrap for 1,000 patterns

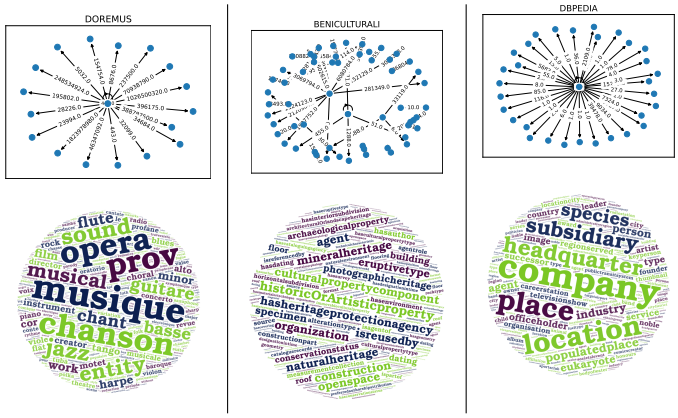

Fig. 7: Visualizing sub-profiles from $N = 10$ patterns of maximal length $M = 5$ drawn by QPLUS

practical element for deducing transaction weights. Building upon the theorems and properties derived from UTU, we unveil effective two-step algorithms tailored for the exploration of vast quantitative databases. Our paper makes significant contributions by providing a robust framework for exploratory data analysis in weighted data, delivering innovative theorems and algorithms for handling large-scale quantitative databases. An extended version of this paper is available at https://arxiv.org/abs/2410.22964.

REFERENCES

[1] R. A. Alva Principe, A. Maurino, M. Palmonari, M. Ciavotta, and B. Spahiu, "Abstat-hd: a scalable tool for profiling very large knowledge graphs," *The VLDB Journal*, pp. 1–26, 2021.

[2] L. Diop, B. Markhoff, and A. Soulet, "Ttprofiler: Types and terms profile building for online cultural heritage knowledge graphs," *J. Comput. Cult. Herit.*, vol. 16, no. 3, aug 2023.

[3] P. Ma, R. Ding, S. Han, and D. Zhang, "Metainsight: Automatic discovery of structured knowledge for exploratory data analysis," in *Proc. SIGMOD*, 2021, p. 1262–1274.

[4] S. Garg, S. Mitra, T. Yu, Y. Gadhia, and A. Kashettiwar, "Reinforced approximate exploratory data analysis," B. Williams, Y. Chen, and J. Neville, Eds. AAAI Press, 2023, pp. 7660–7669.

[5] M. van Leeuwen, *Interactive Data Exploration Using Pattern Mining*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 169–182.

[6] G. Preti, G. De Francisci Morales, and M. Riondato, "Maniacs: Approximate mining of frequent subgraph patterns through sampling," in *Proc. KDD*, 2021, p. 1348–1358.

[7] M. Al Hasan and M. J. Zaki, "Output space sampling for graph patterns," *Proc. of the VLDB Endowment*, vol. 2, no. 1, pp. 730–741, 2009.

[8] M. Boley, C. Lucchese, D. Paurat, and T. Gärtner, "Direct local pattern sampling by efficient two-step random procedures," in *Proc. of the 17th ACM SIGKDD*, 2011, pp. 582–590.

[9] L. Diop, C. T. Diop, A. Giacometti, D. Li Haoyuan, and A. Soulet, "Sequential Pattern Sampling with Norm Constraints," in *IEEE International Conference on Data Mining (ICDM)*, Singapore, Nov. 2018.

[10] L. Diop, "High average-utility itemset sampling under length constraints," in *PAKDD*, 2022, p. 134–148.

[11] H. Tang, J. Wang, and L. Wang, "Mining significant utility discriminative patterns in quantitative databases," *Mathematics*, vol. 11, no. 4, 2023.

[12] H. Yao, H. J. Hamilton, and C. J. Butz, "A foundational approach to mining itemset utilities from databases," in *Proceedings of the Third SIAM International Conference on Data Mining*, 2004, pp. 482–486.

[13] J. C.-W. Lin, T. Li, P. Fournier-Viger, T.-P. Hong, J. Zhan, and M. Voznak, "An efficient algorithm to mine high average-utility itemsets," *Advanced Engineering Informatics*, vol. 30, no. 2, pp. 233 – 243, 2016.