Abstract of "Sampling-based Randomized Algorithms for Big Data Analytics" by Matteo Riondato, Ph.D., Brown University, May 2014.

Analyzing huge datasets becomes prohibitively slow when the dataset does not fit in main memory. Approximations of the results of guaranteed high quality are sufficient for most applications and can be obtained very fast by analyzing a small random part of the data that fits in memory. We study the use of the Vapnik-Chervonenkis dimension theory to analyze the trade-off between the sample size and the quality of the approximation for fundamental problems in knowledge discovery (frequent itemsets), graph analysis (betweenness centrality), and database management (query selectivity).

We show that the sample size to compute a high-quality approximation of the collection of frequent itemsets depends only on the VC-dimension of the problem, which is (tightly) bounded from above by an easy-to-compute characteristic quantity of the dataset. This bound leads to a fast algorithm for mining frequent itemsets that we also adapt to the MapReduce framework for parallel/distributed computation. We exploit similar ideas to avoid the inclusion of false positives in mining results.

The betweenness centrality index of a vertex in a network measures the relative importance of that vertex by counting the fraction of shortest paths going through that vertex. We show that it is possible to compute a high-quality approximation of the betweenness of all the vertices by sampling shortest paths at random. The sample size depends on the VC-dimension of the problem, which is upper bounded by the logarithm of the maximum number of vertices in a shortest path. The tight bound collapses to a constant when there is a unique shortest path between any two vertices.

The selectivity of a database query is the ratio between the size of its output and the product of the sizes of its input tables. Database Management Systems estimate the selectivity of queries for scheduling and optimization purposes. We show that it is possible to bound the VC-dimension of queries in terms of their SQL expressions, and to use this bound to compute a sample of the database that allow much a more accurate estimation of the selectivity than possible using histograms.

Sampling-based Randomized Algorithms for Big Data Analytics

by

Matteo Riondato

Laurea, Università di Padova, Italy, 2007

Laurea Specialistica, Università di Padova, Italy, 2009

Sc. M., Brown University, 2010

A dissertation submitted in partial fulfillment of the

requirements for the Degree of Doctor of Philosophy

in the Department of Computer Science at Brown University

Providence, Rhode Island

May 2014

This dissertation by Matteo Riondato is accepted in its present form by
the Department of Computer Science as satisfying the dissertation requirement
for the degree of Doctor of Philosophy.

Date _____ _____
Eli Upfal, Director

Recommended to the Graduate Council

Date _____ _____
Uğur Çetintemel, Reader

Date _____ _____
Basilis Gidas, Reader
(Applied Mathematics)

Approved by the Graduate Council

Date _____ _____
Peter M. Weber
Dean of the Graduate School

Preface

☑ **TO-DO**: Write acknowledgements

Contents

⋆ Parts of this dissertation appeared in proceedings of conferences or in journals. In particular, Chapter 3 is an extended version of [Riondato and Upfal, 2014], Chapter 4 is an extended version of [Riondato et al., 2012], Chapter 5 is an extended version of [Riondato and Vandin, 2014], Chapter 6 is an extended version of [Riondato and Kornaropoulos, 2014], and Chapter 7 is an extended version of [Riondato et al., 2011].

Chapter 1

Introduction

Thesis statement: Analyzing very large datasets is an expensive computational task. We show that it is possible to obtain high-quality approximations of the results of many analytics tasks by processing only a small random sample of the data. We use the Vapnik-Chervonenkis (VC) dimension to derive the sample size needed to guarantee that the approximation is sufficiently accurate with high probability. This results in very fast algorithms for mining frequent itemsets and association rules, avoiding the inclusion of false positives in mining, computing the betweenness centrality of vertices in a graph, and estimating the selectivity of database queries.

1.1 Motivations

Advancements in retrieval and storage technologies led to the collection of large amounts of data about many aspects of natural and artificial phenomena. Analyzing these datasets to find useful information is a daunting task that requires multiple iterations of data cleaning, modeling, and processing. The cost of the analysis can often be split into two parts. One component of the cost is intrinsic to the complexity of extracting the desired information (e.g., it is proportional to the number of patterns to find, or the number of iterations needed). The second component depends on the size of the dataset to be examined. Smart algorithms can help reducing the first component, while the second seems to be ever increasing as more and more data become available. Indeed the latter may reduce any improvement in the intrinsic cost as the dataset grows. Since many datasets are too large to fit into the main memory of a single machine, analyzing such datasets would require frequent access to disk or even to the network, resulting in excessively long processing times.

The use of random sampling is a natural solution to this issue: by analyzing only a small random sample that fits into main memory there is no need to access the disk or the network. This comes at the inevitable price of only being able to extract an approximation of the results, but the trade-off between the size of the sample and the quality of approximation can be studied using techniques from the theory of probability.

The intuition behind the use of random sampling also allow us to look at a different important issue in data analytics: the statistical validation of the results. Algorithms for analytics often make the implicit assumption that the available dataset constitutes the entire reality and it contains, in some sense, a perfect representation of the phenomena under study. This is indeed not usually the case: not only datasets contain errors due to inherently stochastic collection procedures, but they also naturally contain only a limited amount of information about the phenomena. Indeed a dataset should be seen as a collection of random samples from an unknown data generation process, whose study corresponds to the study of the phenomena. By looking at the dataset this way, it becomes clear that the results of analyzing the dataset are approximated and not "perfect". This leads to the need of validating the results to discriminate between real and spurious information, which was found to be interesting in the dataset only due to the stochastic nature of the data generation process. While the need for statistical validations has since long been acknowledged in the life and physical sciences and it is a central matter of study in statistics, computer science researchers have only recently started to pay attention to it and to develop methods to either validate the results of previously available algorithms or to avoid the inclusion of spurious information in the results.

We study the use of sampling in the following data analytics tasks:

- Frequent itemsets and association rules mining is one of the core tasks of data analytics, requiring to find sets of items in a transactional dataset that appear in more than a given fraction of the transactions, and use this collection to build high-confidence inference rules between sets of items.

- Betweenness centrality is a measure of the importance of a vertex in a graph, corresponding to the fraction of shortest paths going through that vertex. By ranking vertices according to their betweenness centrality one can find which ones are most important, for example for marketing purposes.

- The selectivity of a database query is the ratio between the size of its output and the product

of the sizes of the tables in input. Database management systems estimate the selectivity of queries to make informed scheduling decisions.

We choose to focus on important tasks from different areas of data analytics, rather than a single task or different tasks from the same area. This is motivated by our goal of showing that random sampling can be used to compute good approximations for a variety of tasks. Moreover, these tasks share some common aspects: they involve the analysis of very large datasets and they involve the estimation of a measure of interest (e.g., the frequency) for many objects (e.g., all itemsets). These aspects are common in many other tasks.

As we said, there exists many mathematical tools that can be used to compute a sample size sufficient to extract a high-quality approximation of the results: Chernoff/Hoeffding bounds, martingales, tail bound on polynomials of random variables, and more [Alon and Spencer, 2008; Dubhashi and Panconesi, 2009; Mitzenmacher and Upfal, 2005]. We choose to focus on VC-dimension for the following reasons:

- Classical probabilistic tools like the Azuma inequality work by bounding the probability that the measure of interest for a single object deviates from its expectation by more than a given quantity. One must then apply the union bound to get simultaneous guarantees for all the objects. This results in a sample size that depends on the logarithm of the number of objects, which may be still too much and too loose. Previous works explored the use of these tools to solve the problems we are interested in and indeed suffered from this drawback. On the other hand, results related to VC-dimension give sample sizes that only depend on this combinatorial quantity, which can be very small and independent from the number of objects.
- The use of VC-dimension is not limited to a specific problem or setting. The results and techniques related to VC-dimension are widely applicable and can be used for many different problems. Moreover, no assumption is made on the distribution of the measure of interest, or on the availability of any previous knowledge about the data.

1.2 Overview of contributions

This dissertation presents sampling-based algorithms for a number of data analytics tasks: frequent itemsets and association rules mining, betweenness centrality approximation, and datatabase query

selectivity estimation. We use the Vapnik-Chervonenkis dimension and related results from the field of statistical learning theory (Ch. 2) to analyze our algorithms . This leads to significant reductions in the sample sizes needed to obtain high-quality approximations of the results.

Mining frequent itemsets and association rules. We show that the VC-dimension of the problem of mining frequent itemsets and association rules is tightly bounded by an easy-to-compute characteristic quantity of the dataset. This allows us to derive the smallest known bound to the sample size needed to compute very accurate approximations of the collections of patterns (Ch. 3). The combination of the theoretical guarantees offered by these results with the computational power and scalability of MapReduce allows us to develop PARMA, a fast distributed algorithm that mines many small samples in parallel to achieve higher confidence and accuracy in the output collection of patterns. PARMA scales much better than existing solutions as the dataset grows and uses all the available computational resources (Ch. 4).

Statistical validation. A common issue when mining frequent itemsets is the inclusion of false positives, i.e., of itemsets that are only frequent by chance. We develop a method that computes a bound to the VC-dimension of a collection of itemsets by solving a variant of the Set-Union Knapsack Problem. This allows us to to find a frequency threshold such that the collection of frequent itemsets with respect to this threshold will not include false positives with high probability (Ch. 5).

Betweenness centrality. We show that it is possible to compute very accurate estimations of the betweenness centrality (and many of its variants) of all vertices by only performing a limited number of shortest paths computation, chosen at random. The sample size depends on the VC-dimension of the problem, which we show to be bounded by the logarithm of the number of vertices in the longest shortest path (i.e., by the logarithm of the diameter in an unweighted graph). The bound is tight and interestingly collapses to a small constant when there is a unique shortest path between every pair of connected vertices. Our results also allow us to present a tighter analysis of an existing sampling-based algorithm for betweenness centrality estimation (Ch. 6).

Query selectivity estimation. We show that the VC-dimension of classes of database queries is bounded by the complexity of their SQL expression, i.e., by the maximum number of joins and selection predicates involved in the queries. Leveraging on this bound, it is possible to obtain

very accurate estimations of the selectivities by running the queries on a very small sample of the database.

Chapter 2

The Vapnik-Chervonenkis Dimension

In this dissertation we explore the use of random sampling to develop fast and efficient algorithms for data analytics problems. The use of sampling is not new in this area but previously presented algorithms were severely limited by their use of classic probability tools like the Chernoff bound, the Hoeffding bound, and the union bound [Mitzenmacher and Upfal, 2005] to derive the sample size necessary to obtain approximations of (probabilistically) guaranteed quality. Instead, we show that it is possible to obtain much smaller sample sizes, and therefore much more efficient algorithms, by using concepts and results from *statistical learning theory* [Vapnik, 1998, 1999], in particular those involving the *Vapnik-Chervonenkis (VC) dimension* of the problem.

2.1 Use of VC-Dimension in computer science

VC-Dimension was first introduced in a seminal article [Vapnik and Chervonenkis, 1971] on the convergence of empirical averages to their expectations, but it was only with the work of Haussler and Welzl [1986] and Blumer et al. [1989] that it was applied to the field of learning. It became a core component of Valiant's Probably Approximately Correct (PAC) Framework [Kearns and Vazirani, 1994] and has enjoyed enormous success and application in the fields of computational geometry [Chazelle, 2000; Matoušek, 2002] and machine learning [Anthony and Bartlett, 1999; Devroye et al., 1996]. Boucheron et al. [2005] present a good survey and many recent developments. Other applications include database management and graph algorithms. In the former, it was used in the

context of constraint databases to compute good approximations of aggregate operators [Benedikt and Libkin, 2002]. VC-dimension-related results were also recently applied in the field of database privacy by Blum et al. [2008] to show a bound on the number of queries needed for an attacker to learn a private concept in a database. Gross-Amblard [2011] showed that content with unbounded VC-dimension can not be watermarked for privacy purposes. In the graph algorithms literature, VC-Dimension has been used to develop algorithms to efficiently detect network failures [Kleinberg, 2003; Kleinberg et al., 2008], balanced separators [Feige and Mahdian, 2006], events in a sensor networks [Gandhi et al., 2010], and compute approximations of the shortest paths [Abraham et al., 2011].

We outline here only the definitions and results that we use throughout the dissertation. We refer the reader to the works of Alon and Spencer [2008, Sect. 14.4], Boucheron et al. [2005, Sect. 3], Chazelle [2000, Chap. 4], Devroye et al. [1996, Sect. 12.4], Mohri et al. [2012, Chap. 3], and Vapnik [1998, 1999] for more details on the VC-dimension theory.

2.2 Range spaces, VC-dimension, and sampling

A *range space* is a pair $(D, \mathcal{R})$ where D is a (finite or infinite) domain and $\mathcal{R}$ is a (finite or infinite) family of subsets of D. The members of D are called *points* and those of R are called *ranges*. The Vapnik-Chernovenkis (VC) Dimension of $(D, \mathcal{R})$ is a measure of the *complexity* or *expressiveness* of $\mathcal{R}$ [Vapnik and Chervonenkis, 1971]. Knowning (an upper bound) to the VC-dimension of a range space can be very useful: if we define a probability distribution ν over D, then a finite upper bound to the VC-dimension of $(D, \mathcal{R})$ implies a bound to the number of random samples from ν required to approximate the probability $\nu(R) = \sum_{r \in R} \nu(r)$ of each range R simultaneously using the *empirical average* of $\nu(R)$ as estimator.

Given $A \subset D$, The *projection* of $\mathcal{R}$ on A is defined as $P_{\mathcal{R}}(A) = \{R \cap A \ : \ R \in \mathcal{R}\}$. If $P_{\mathcal{R}}(A) = 2^A$, then A is said to be *shattered by* $\mathcal{R}$.

Definition 1. Given a set $B \subseteq D$, the *empirical Vapnik-Chervonenkis (VC) dimension of* $\mathcal{R}$ *on* B, denoted as $\mathsf{EVC}(\mathcal{R}, B)$ is the cardinality of the *largest* subset of B that is shattered by $\mathcal{R}$. If there are arbitrary large shattered subsets, then $\mathsf{EVC}() = \infty$. The *VC-dimension of* $(D, \mathcal{R})$ is defined as $\mathsf{VC}\left((D, \mathcal{R})\right) = \mathsf{EVC}(\mathcal{R}, D)$.

For any range space, we have that its VC-dimension cannot be greater than $d = \lfloor \log_2 |\mathcal{R}| \rfloor$ since for any $t > d$, one would need 2^t ranges to shatter a set of size t.

A range space $(D, \mathcal{R})$ with an infinite set of points D and an infinite family of ranges $\mathcal{R}$ can have a *finite* VC-dimension. A simple example is the family of open intervals in $\mathbb{R}$ (i.e., $D = \mathbb{R}$ and $\mathcal{R}$ contains all the half-closed intervals $[a, +\infty)$ and $(-\infty, a)$ for any $a \in \mathbb{R}$). Let $A = \{x, y, z\}$ be the set of three points such that $x < y < z$. There is no interval $R \in \mathcal{R}$ such that $R \cap A = \{x, z\}$ so the VC-dimension of this range space is less than 3. This result can be generalized to higher dimensions.

Lemma 1 (Lemma 10.3.1 [Matoušek, 2002])**.** *The VC-Dimension of the range space* $(\mathbb{R}^d, \mathcal{R})$*, where* $\mathcal{R}$ *is the set of all half-spaces in* $\mathbb{R}^d$ *equals* $d + 1$.

Another example of range space and its VC-dimension is shown in Fig. 2.1.

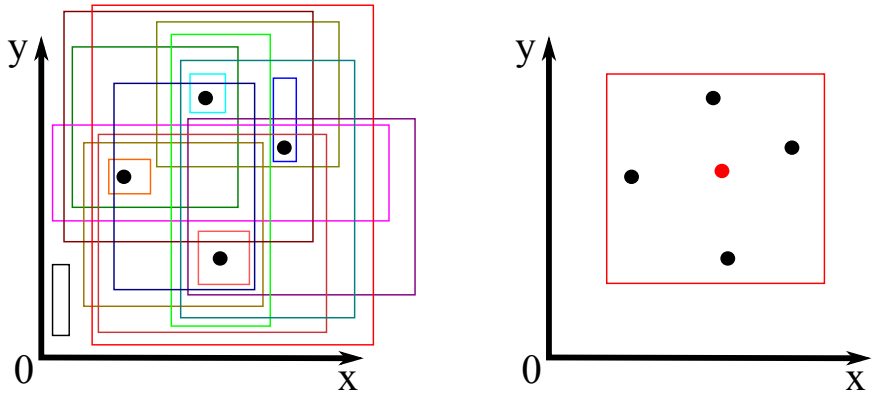


Figure 2.1: Example of range space and VC-dimension. The space of points is the plane $\mathbb{R}^2$ and the set $\mathcal{R}$ of ranges is the set of all *axis-aligned rectangles*. The figure on the left shows graphically that it is possible to shatter a set of four points using 16 rectangles. On the right instead, one can see that it is impossible to shatter five points, as, for any choice of the five points, there will always be one (the red point in the figure) that is internal to the convex hull of the other four, so it would be impossible to find an axis-aligned rectangle containing the four points but not the internal one. Hence $\mathsf{VC}((\mathbb{R}^2, \mathcal{R})) = 4$.

A classic example of a range space with infinite VC-dimension is the space of sine functions. Let $D = \mathbb{R}^2$ and, for any $\omega \in \mathbb{R}$, let A_ω be the subset of $\mathbb{R}^2$ that lies *above* the curve $\sin(\omega t)$, for $t \in \mathbb{R}$. It is possible to shatter any set S of points by choosing an appropriate ω_A for each subset $A \subseteq S$.

Let ν be a probability distribution on the points of D, and let $X_1^k = (X_1, \ldots, X_k)$ be a bag of elements from D. For any subset $A \subseteq D$, we define the function

$$\nu_{X_1^k}(A) = \frac{1}{k} \sum_{j=1}^{k} \mathbb{1}_A(X_j),$$

where $\mathbb{1}_A$ is the indicator function for the set A. When X_1^k is a collection of *independent samples from* ν, then $\nu_{X_1^k}(A)$ is known as the *empirical average* of $\nu(A) = \sum_{a \in A} \nu(a)$ on X_1^k and is an *unbiased*

estimator for $\nu(A)$ (i.e., $\mathbb{E}[\nu_{X_1^k}(A)] = \nu(A)$). One of the main applications of (empirical) VC-dimension is in giving bounds to the number of samples from ν needed to *simultaneously* approximate the probabilities $\nu(A)$ of all $A \in \mathcal{R}$ using their empirical averages, in the following sense.

Definition 2. Let $(D, \mathcal{R})$ be a range space and ν be a probability distribution on D. For $\varepsilon \in (0,1)$, an *ε-approximation to* $(D, \mathcal{R}, \nu)$ is a bag S of elements of D such that

$$\sup_{A \in \mathcal{R}} |\nu(A) - \nu_S(A)| \leq \varepsilon \ . \tag{2.1}$$

If D is *finite* and ν is the *uniform* distribution on D, then $\nu(A) = |A|/|D|$, and (2.1) is equivalent to

$$\left| \frac{|A|}{|D|} - \frac{|A \cap S|}{|S|} \right| \leq \varepsilon, \forall A \in \mathcal{R}.$$

Indeed this will be often but not always the case in later chapters, and we will simplify the notation by dropping ν and saying that S is an ε-approximation to $(D, \mathcal{R})$.

A great result by Vapnik and Chervonenkis [1971] showed that it is possible to compute a ε-approximation with a limited number samples from ν, provided an upper bound to the VC-dimension of $(D, \mathcal{R})$ or to its empirical VC-dimension on a subset of D is known. This result is one of the reasons why VC-dimension gained fundamental importance in the PAC framework [Kearns and Vazirani, 1994]. The original bound to the number of samples was improved multiple times. Here we present the most recent improvement.

Theorem 1 (Thm. 2.12 [Har-Peled and Sharir, 2011], see also [Li et al., 2001])**.** *Let $(D, \mathcal{R})$ be a range space with $\mathsf{VC}(\mathcal{R}) \leq d$, and let ν be a distribution on D. Given $\varepsilon, \delta \in (0,1)$, let*

$$\ell = \frac{c}{\varepsilon^2} \left(d + \log \frac{1}{\delta} \right) \tag{2.2}$$

where c is an universal constant. Then, a bag of ℓ elements of D sampled independently according to ν is an ε-approximation to $(\mathcal{R}, \nu)$ with probability at least $1 - \delta$.

The constant c is approximately 0.5 and it is *universal*, i.e., it does not depend on any parameter [Löffler and Phillips, 2009]. It is also interesting to note that, if $|D|$ is finite, then an ε-approximation of size $O(\frac{d}{\varepsilon^2} \log \frac{d}{\varepsilon})$ can be built *deterministically* in time $O(d^{3d}(\frac{1}{\varepsilon^2} \log \frac{d}{\varepsilon})^d |D|)$ [Chazelle, 2000].

Throughout this dissertation we assume the sample to be drawn *with* replacement if the sample size is smaller than the size $|D|$ of the domain, (otherwise the sample is exactly D).

To understand the importance of Thm. 1, consider the following simple example. Assume that $\mathcal{R}$ contains a finite number of ranges and let ν be the *uniform* distribution on D, and S be a sample of points drawn uniformly and independently at random (i.e., according to ν) from D. We can compute the sample size $|S|$ needed to obtain an ε-approximation with probability at least $1-\delta$ by using the Chernoff bound and the union bound [Mitzenmacher and Upfal, 2005]. Indeed, using the union bound, we have that

$$\Pr\left(\exists A \in \mathcal{R} \ : \ \left|\nu(A) - \frac{|A \cap S|}{|S|}\right| > \varepsilon\right) \leq \sum_{A \in \mathcal{R}} \Pr\left(\left|\nu(A) - \frac{|A \cap S|}{|S|}\right| > \varepsilon\right)$$

If the right side is bounded by δ, then S is a ε-approximation for $(D, \mathcal{R}, \nu)$. A sufficient condition for this is that, for any $A \in \mathcal{R}$,

$$\Pr\left(\left|\nu(A) - \frac{|A \cap S|}{|S|}\right| > \varepsilon\right) \leq \frac{\delta}{|\mathcal{R}|}$$

The quantity $|A \cap S|$ is a random variable with binomial distribution $B(|S|, \nu(A))$. Then we can use the Chernoff bound and obtain

$$\Pr\left(\left|\nu(A) - \frac{|A \cap S|}{|S|}\right| > \varepsilon\right) \leq 2e^{-2|S|\varepsilon^2}$$

which is smaller than $\delta/|\mathcal{R}|$ for

$$|S| \geq \frac{1}{2\varepsilon^2}\left(\ln|\mathcal{R}| + \ln 2 + \ln\frac{1}{\delta}\right).$$

Comparing this quantity with (2.2) clearly shows the multiple advantages of VC-dimension. Firstly, the sample size suggested by (2.2) is smaller than the above as soon as the (upper bound to the) VC-dimension of $(D, \mathcal{R})$ is smaller than $\ln(2\mathcal{R})$ (note that the above example cannot be used when $\mathcal{R}$ has infinite size). Secondly, Thm. 1 holds for *any* distribution ν and no assumption are made on it or any of its moments (e.g., on its variance). It is important to mention that if $\mathsf{VC}\left((D, \mathcal{R})\right)$ and/or the upper bound d do not depend on $|D|$ or on $|\mathcal{R}|$ neither does the sample size presented in Thm. 1 (nor the ones in Thm. 2 and Thm. 3 which we show next). We use this property in the following chapters to develop efficient sampling-based randomize algorithms for important problems in data mining.

It is possible to build an ε-approximation even when only an upper bound to the empirical

VC-Dimension is available.

Theorem 2 (Sect. 3 [Boucheron et al., 2005]). *Let $(D, \mathcal{R})$ be a range space, and let ν be a distribution on D. Let S be a bag of ℓ points from D sampled independently according to ν. Let d be an integer such that $\mathsf{EVC}(\mathcal{R}, X_1^\ell) \leq d$. Given $\delta \in (0,1)$, let*

$$\varepsilon = 2\sqrt{\frac{2d\log(\ell+1)}{\ell}} + \sqrt{\frac{2\log\frac{2}{\delta}}{\ell}}. \tag{2.3}$$

Then S is a ε-approximation for $(\mathcal{R}, \nu)$ with probability at least $1-\delta$.

One can define and compute stronger approximations with *relative* guarantees (rather than additive) as follows.

Definition 3. Let $(D, \mathcal{R})$ be a range space and ν be a probability distribution on D. For $p, \varepsilon \in (0,1)$, a *relative (p,ε)-approximation to $(D, \mathcal{R}, \nu)$* is a bag S of elements from D such that

- For any $A \in \mathcal{R}$ such that $\nu(A) \geq p$, we have

$$|\nu(A) - \nu_S(A)| \leq \varepsilon\nu(A) \ .$$

- For any $B \in \mathcal{R}$ such that $\nu(B) < p$, we have $\nu_S(B) \leq (1+\varepsilon)p$.

Theorem 3 (Thm. 2.11 [Har-Peled and Sharir, 2011]). *Let $(D, \mathcal{R})$ be a range space with $\mathsf{VC}((D, \mathcal{R})) \leq d$, and let ν be a distribution on D. Given $\varepsilon, \delta, p \in (0,1)$, let*

$$\ell \geq \frac{c'}{\varepsilon^2 p}\left(d\log\frac{1}{p} + \log\frac{1}{\delta}\right) \tag{2.4}$$

where c' is an universal constant. Then a bag of ℓ points from D sampled according to ν is a relative (p,ε)-approximation to $(D, \mathcal{R}, \nu)$ with probability at least $1-\delta$.

Up to a constant factor, the bounds presented in Thm. 1 and Thm. 3 are tight [Li et al., 2001, Thm. 5].

In Sect. 7.3 we use the following bound which is an extension of [Alon and Spencer, 2008, Corol. 14.4.3] to arbitrarily combinations of set operations. We present it here because it is a general result on VC-dimension.

Lemma 2. *Let $(D, \mathcal{R})$ be a range space of VC-dimension $d \geq 2$ and let $(D, \mathcal{R}_h)$ be the range space on D in which $\mathcal{R}_h$ includes all possible combinations of union and intersections of h members of $\mathcal{R}$. Then $\mathsf{VC}((D, \mathcal{R}_h)) \leq 3dh\log(dh)$.*

Proof. Let A be an arbitrarily subset of cardinality n of D. From [Alon and Spencer, 2008, Coroll. 14.4.2], we have that $|P_{\mathcal{R}}(A)| \leq n^d$. There are

$$\binom{|P_{\mathcal{R}}(A)|}{h} \leq n^{dh}$$

possible choices of h members of $P_{\mathcal{R}}(A)$, and there are no more than $h!2^{h-1}C_{h-1} \leq h^{2h}$ Boolean combinations using unions and intersections of the h sets, where C_{h-1} is the $(h-1)^{\text{th}}$ Catalan number ($C_i = \frac{1}{i+1}\binom{2i}{i}$). If $2^n > h^{2h}n^{dh} \geq |P_{\mathcal{R}_h}(A)|$ then A cannot be shattered. This inequality holds for $n \geq 3dh\log(dh)$. To see this, consider the fact that clearly $2^n \geq h^{2h}n^{dh}$ for sufficiently large n, so it suffices to show that $n = 3dh\log(dh)$ is sufficiently large. Next observe that the function $f(x) = 2\log x - \log(3x\log x)$ is positive for $x \geq 2$ since it is positive for $x = 2$ and it is easy to see that the derivative $f'(x)$ is positive for $x \geq 2$. It immediately follows that $3\log(dh) > \log(dh) + \log(3dh\log(dh)$ for $d \geq 2$ and $h \geq 1$. Since $dh\log(dh) \geq 2h\log(h)$, we have $3dh\log(dh) > 2h\log(3dh\log(dh))$, which proves the result. □

2.3 Computational considerations

An upper bound to the VC-dimension of the range space is needed in order to apply Thm. 1, Thm. 2, or Thm. 3. It is natural to ask whether, rather than a bound, one could compute the exact VC-dimension of a range space. Papadimitriou and Yannakakis [1996] characterized the complexity of computing the VC-dimension. Although in NP, the problem is most likely not NP-complete, but can it be solved in polynomial time? Papadimitriou and Yannakakis [1996] defined a new complexity class LOGNP and show that the decision version of the problem of computing the VC-dimension is LOGNP-complete. Schäfer [1999] showed that in general one can not approximate the problem up to a constant factor in polynomial time, unless P=NP. Mossel and Umans [2001] gave a refined characterization of the approximability of the problem.

Vapnik et al. [1994] and Shao et al. [2000] presented and refined an experimental procedure to estimate the VC-dimension of a learning machine, and McDonald et al. [2011] gave concentration results for such an estimate. This procedure is mostly of theoretical interests, as it is computationally very expensive. It would not be practical in settings like the one we study in this dissertation, as it would defeat the purpose of using sampling to speed up the analysis of very large datasets.

Chapter 3

Mining Association Rules and Frequent Itemsets

In this chapter we start presenting our contributions to the problem of extracting interesting collections of patterns from very large transactional dataset.

The discovery of Frequent Itemsets and Association Rules is a fundamental computational primitive with application in data mining (market basket analysis), databases (histogram construction), networking (heavy hitters) and more [Han et al., 2007, Sect. 5]. The computational problem is defined in the general setting of a transactional dataset – a collection of transactions where each transaction is a set of items. With datasets increasing both in size and complexity, the computation for FIM faces scalability challenges in both space and time. Depending on the particular application, one is interested in finding all itemsets with frequency greater or equal to a user defined threshold (FIs), identifying the K most frequent itemsets (top-K), or computing all association rules (ARs) with user defined minimum support and confidence level (see Sect. 3.4.4 and 3.4.5 for additional criteria). The cost of algorithms for extracting the FIs and the ARs can be split in two independent components: the *scanning* cost and the *mining* cost. The scanning cost includes all operations that directly handle the transactions in the dataset, and scales with the *size* of the dataset, i.e., the number of such transactions. Examples include the scanning of the dataset to build the FP-Tree in FP-Growth [Han et al., 2000] or to compute the actual frequencies of candidate frequent itemsets in APriori [Agrawal et al., 1993]. The mining cost refers to the operations in derived data structures

This chapter is an extended version of a work that originally appeared in ACM Transaction of Knowledge Discovery in Data as [Riondato and Upfal, 2014].

and does not require access to the dataset. Examples include the operations performed on the FP-Tree once it has been generated, and the creation of candidate itemsets of length $i + 1$ at the end of phase i in APriori. This cost scales with the *complexity* of the dataset, i.e., the number of items, the number and distribution of frequent itemsets, and the underlying process that generated the transactions. It also depends on parameters given to the algorithm, such as the desired frequency threshold.

A typical exact algorithm scans the entire dataset, possibly multiple times, and stores intermediate counts of a large number of possible frequent itemsets candidates [Agrawal and Srikant, 1994; Han et al., 2000]. For large datasets that do not fit in main memory, this can be prohibitively expensive. We are interested in reducing the scanning cost of algorithms by using random sampling to compute high quality approximations of the collections of FIs and ARs, which are usually sufficient for most practical applications. It is reasonable to focus on the scanning cost as in many practical settings the process generating the data changes very slowly or not at all, especially when compared to the data generation rate, therefore the number and frequency distribution of the frequent itemsets grows much slower than the size of the dataset. For example, the number of items available on the catalog of an e-commerce website grows much slower than the number of purchases by customers, each of which corresponds to a transaction. Therefore the scanning component grows faster than the mining one, and soon becomes dominant.

Indeed, a number of recent works (see Sect. 3.1 for more details) explored the application of sampling for approximate solutions to these problems. However, the efficiency and practicality of the sampling approach depends on a tight relation between the size of the sample and the quality of the resulting approximation. Previous works do not provide satisfactory solutions to this problem.

The technical difficulty in analyzing any sampling technique for frequent itemset discovery problems is that a-priori any subset of items can be among the most frequent ones, and the number of subsets is exponential in the number of distinct items appearing in the dataset. A standard analysis begins with a bound on the probability that a given itemset is either over or under represented in the sample. Such bound is easy to obtain using a large deviation bound such as the Chernoff bound or the Central Limit theorem [Mitzenmacher and Upfal, 2005]. The difficulty is in combining the bounds for individual itemsets into a global bound that holds simultaneously for all the itemsets. A simple application of the union bound vastly overestimates the error probability because of the large number of possible itemsets, a large fraction of which may not be present in the dataset and therefore

should not be considered. More sophisticated techniques, developed in recent works [Chakaravarthy et al., 2009; Chuang et al., 2005; Pietracaprina et al., 2010], give better bounds only in limited cases. A loose bound on the required sample size for achieving the user defined performance guarantees, decreases the gain obtained from the use of sampling.

We circumvent this issue by applying the VC-dimension theory to frequent itemsets problems by viewing the presence of an itemset in a transaction as the outcome of an indicator function associated with the itemset. The major theoretical contributions of our work are a complete characterization of the VC-dimension of the range space associated with a dataset, and a tight bound to this quantity. We prove that the VC-dimension is upper bounded by a characteristic quantity of the dataset which we call *d-index*. The d-index is the maximum integer d such that the dataset contains at least d different transactions of length at least d such that no one of them is a subset of or equal to another in the considered set of transactions (see Def. 9). We show that this bound is tight by demonstrating a large class of datasets with a VC-dimension that matches the bound. Computing the d-index can be done in polynomial time but it requires multiple scans of the dataset. We show how to compute an upper bound to the d-index with a single linear scan of the dataset in an online greedy fashion.

The VC-dimension approach provides a unified tool for analyzing the various frequent itemsets and association rules problems (i.e., the market basket analysis tasks). We use it to prove tight bounds on the required sample size for extracting FIs with a minimum frequency threshold, for mining the top-K FIs, and for computing the collection of ARs with minimum frequency and "interestingness" thresholds, where the interestingness can be expressed in terms of confidence, leverage, lift, or other measure. Furthermore, we compute bounds for both absolute and relative approximations (see Sect. 3.2 for definitions) and our results extend to a variety of other measures proposed in the literature (see Sect. 3.4.4). We show that high quality approximations can be obtained by mining a very small random sample of the dataset. Table 3.1 compares our technique to the best previously known results for the various problems (see Sect. 3.2 for definitions). Our bounds, which are linear in the VC-dimension associated with the dataset, are consistently smaller than previous results and less dependent on other parameters of the problem such as the minimum frequency threshold and the dataset size. An extensive experimental evaluation demonstrates the advantage of our technique in practice.

We are the first to provide a characterization and an explicit bound for the VC-dimension of the range space associated with a dataset and to apply the result to the extraction of FIs and ARs from

random sample of the dataset. We believe that this connection with statistical learning theory can be furtherly exploited in other data mining problems.

Table 3.1: Required sample sizes (as number of transactions) for various approximations to FIs and ARs as functions of the VC-dimension v, the maximum transaction length Δ, the number of items $|\mathcal{I}|$, the accuracy ε, the failure probability δ, the minimum frequency θ, and the minimum confidence γ. Note that $v \leq \Delta \leq |\mathcal{I}|$ (but $v < |\mathcal{I}|$). The constant c and c' are absolute, with $c \leq 0.5$. See Sect. 3.4.4 for the sample sizes for approximations of the collection of association rules according to interestingness measures other than confidence.

Task/Approx.	This work	Best previous work
FIs/abs.	$\frac{4c}{\varepsilon^2}\left(v + \log\frac{1}{\delta}\right)$	$O\left(\frac{1}{\varepsilon^2}\left(\lvert\mathcal{I}\rvert + \log\frac{1}{\delta}\right)\right)^{\dagger}$
FIs/rel.	$\frac{4(2+\varepsilon)c}{\varepsilon^2(2-\varepsilon)\theta}\left(v\log\frac{2+\varepsilon}{\theta(2-\varepsilon)} + \log\frac{1}{\delta}\right)$	$\frac{24}{\varepsilon^2(1-\varepsilon)\theta}\left(\Delta + 5 + \log\frac{4}{(1-\varepsilon)\theta\delta}\right)^{\ddagger}$
top-K FIs/abs.	$\frac{16c}{\varepsilon^2}\left(v + \log\frac{1}{\delta}\right)$	$O\left(\frac{1}{\varepsilon^2}\left(\lvert\mathcal{I}\rvert + \log\frac{1}{\delta}\right)\right)^{\S}$
top-K FIs/rel.	$\frac{4(2+\varepsilon)c'}{\varepsilon^2(2-\varepsilon)\theta}\left(v\log\frac{2+\varepsilon}{\theta(2-\varepsilon)} + \log\frac{1}{\delta}\right)$	not available
ARs/abs.	$O\left(\frac{(1+\varepsilon)}{\varepsilon^2(1-\varepsilon)\theta}\left(v\log\frac{1+\varepsilon}{\theta(1-\varepsilon)} + \log\frac{1}{\delta}\right)\right)$	not available
ARs/rel.	$\frac{16c'(4+\varepsilon)}{\varepsilon^2(4-\varepsilon)\theta}\left(v\log\frac{4+\varepsilon}{\theta(4-\varepsilon)} + \log\frac{1}{\delta}\right)$	$\frac{48}{\varepsilon^2(1-\varepsilon)\theta}\left(\Delta + 5 + \log\frac{4}{(1-\varepsilon)\theta\delta}\right)^{\P}$

† [Jia and Lu, 2005; Li and Gopalan, 2005; Toivonen, 1996; Zhang et al., 2003]
‡ [Chakaravarthy et al., 2009]
§ [Pietracaprina et al., 2010; Scheffer and Wrobel, 2002]
¶ [Chakaravarthy et al., 2009]

Outline. We review relevant previous work in Sect. 3.1. In Sect. 3.2 we formally define the problem and our goals, and introduce definitions and lemmas used in the analysis. The main part of the analysis with derivation of a strict bound to the VC-dimension of association rules is presented in Sect. 3.3, while our algorithms and sample sizes for mining FIs, top-K FIs, and association rules through sampling are in Sect. 3.4. Section 3.5 contains an extensive experimental evaluation of our techniques. A discussion of our results and the conclusions can be found in Sect. 3.6.

3.1 Related work

Agrawal et al. [1993] introduced the problem of mining association rules in the basket data model, formalizing a fundamental task of information extraction in large datasets. Almost any known algorithm for the problem starts by solving a FIs problem and then generate the association rules

implied by these frequent itemsets. Agrawal and Srikant [1994] presented *Apriori*, the most well-known algorithm for mining FIs, and *FastGenRules* for computing association rules from a set of itemsets. Various ideas for improving the efficiency of FIs and ARs algorithms have been studied, and we refer the reader to the survey by Ceglar and Roddick [2006] for a good presentation of recent contributions. However, the running times of all known algorithms heavily depend on the size of the dataset.

Mannila et al. [1994] were the first to propose the use of sampling to efficiently identify the collection of FIs, presenting some empirical results to validate the intuition. Toivonen [1996] presents an algorithm that, by mining a random sample of the dataset, builds a candidate set of frequent itemsets which contains all the frequent itemsets with a probability that depends on the sample size. There are no guarantees that all itemsets in the candidate set are frequent, but the set of candidates can be used to efficiently identify the set of frequent itemsets with at most two passes over the entire dataset. This work also suggests a bound on the sample size sufficient to ensure that the frequencies of itemsets in the sample are close to their real one. The analysis uses Chernoff bounds and the union bound. The major drawback of this sample size is that it depends linearly on the number of individual items appearing in the dataset.

Zaki et al. [1997] show that static sampling is an efficient way to mine a dataset, but choosing the sample size using Chernoff bounds is too conservative, in the sense that it is possible to obtain the same accuracy and confidence in the approximate results at smaller sizes than what the theoretical analysis proves.

Other works tried to improve the bound to the sample size by using different techniques from statistics and probability theory like the central limit theorem [Jia and Lu, 2005; Li and Gopalan, 2005; Zhang et al., 2003] or hybrid Chernoff bounds [Zhao et al., 2006].

Since theoretically-derived bounds to the sample size were too loose to be useful, a corpus of works applied progressive sampling to extract FIs [Brönnimann et al., 2003; Chandra and Bhaskar, 2011; Chen et al., 2002, 2011; Chuang et al., 2005; Hu and Yu, 2006; Hwang and Kim, 2006; Jia and Gao, 2005; John and Langley, 1996; Mahafzah et al., 2009; Parthasarathy, 2002; Wang et al., 2005b]. Progressive sampling algorithms work by selecting a random sample and then trimming or enriching it by removing or adding new sampled transactions according to a heuristic or a self-similarity measure that is fast to evaluate, until a suitable stopping condition is satisfied. The major downside of this approach is that it offers no guarantees on the quality of the obtained results.

Another approach to estimating the required sample size is presented by Chuang et al. [2008]. The authors give an algorithm that studies the distribution of frequencies of the itemsets and uses this information to fix a sample size for mining frequent itemsets, but without offering any theoretical guarantee.

A recent work by Chakaravarthy et al. [2009] gives the first analytical bound on a sample size that is linear in the length of the longest transaction, rather than in the number of items in the dataset. This work is also the first to present an algorithm that uses a random sample of the dataset to mine approximated solutions to the ARs problem with quality guarantees. No experimental evaluation of their methods is presented, and they do not address the top-K FIs problem. Our approach gives better bounds for the problems studied in [Chakaravarthy et al., 2009] and applies to related problems such as the discovery of top-K FIs and absolute approximations.

Extracting the collection of top-K frequent itemsets is a more difficult task since the corresponding minimum frequency threshold is not known in advance [Cheung and Fu, 2004; Fu et al., 2000]. Some works solved the problem by looking at *closed* top-K frequent itemsets, a concise representation of the collection [Pietracaprina and Vandin, 2007; Wang et al., 2005a], but they suffers from the same scalability problems as the algorithms for exactly mining FIs with a fixed minimum frequency threshold.

Previous works that used sampling to approximation the collection of top-K FIs [Pietracaprina et al., 2010; Scheffer and Wrobel, 2002] used progressive sampling. Both works provide (similar) theoretical guarantees on the quality of the approximation. What is more interesting to us, both works present a theoretical upper bound to the sample size needed to compute such an approximation. The size depended linearly on the number of items. In contrast, our results give a sample size that only in the worst case is linear in the number of items but can be (and is, in practical cases) much less than that, depending on the dataset, a flexibility not provided by previous contributions. Sampling is used by Vasudevan and Vojonović [2009] to extract an approximation of the top-K frequent individual *items* from a sequence of items, which contains no item whose actual frequency is less than $f_K - \varepsilon$ for a fixed $0 < \varepsilon < 1$, where f_K is the *actual* frequency of the K-th most frequent item. They derive a sample size sufficient to achieve this result, but they assume the knowledge of f_K, which is rarely the case. An empirical sequential method can be used to estimate the right sample size. Moreover, the results cannot be directly extended to the mining of top-K frequent item(set)s from datasets of transactions with length greater than one.

To our knowledge, ours is the first application of VC-dimension to knowledge discovery.

3.2 Preliminaries

This section introduces basic definitions and properties that will be used in later sections.

A *dataset* $\mathcal{D}$ is a collection of *transactions*, where each transaction τ is a subset of a ground set $\mathcal{I}$[1] There can be multiple identical transactions in $\mathcal{D}$. Elements of $\mathcal{I}$ are called *items* and subsets of $\mathcal{I}$ are called *itemsets*. Let $|\tau|$ denote the number of items in transaction τ, which we call the *length* of τ. Given an itemset $A \subseteq \mathcal{I}$, the *support set* of A, denoted as $T_{\mathcal{D}}(A)$, is the set of transactions in $\mathcal{D}$ that contain A. The *support* of A, $s_{\mathcal{D}}(A) = |T_{\mathcal{D}}(A)|$, is the number of transaction in $\mathcal{D}$ that contains A, and the *frequency* of A, $f_{\mathcal{D}}(A) = |T_{\mathcal{D}}(A)|/|\mathcal{D}|$, is the fraction of transactions in $\mathcal{D}$ that contain A.

Definition 4. Given a *minimum frequency threshold* θ, $0 < \theta \leq 1$, the *FIs mining task with respect to* θ is finding all itemsets with frequency $\geq \theta$, i.e., the set

$$\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta) = \{(A, f_{\mathcal{D}}(A)) \ : \ A \subseteq \mathcal{I} \text{ and } f_{\mathcal{D}}(A) \geq \theta\}\,.$$

To define the collection of top-K FIs, we assume a fixed *canonical ordering* of the itemsets in $2^{\mathcal{I}}$ by decreasing frequency in $\mathcal{D}$, with ties broken arbitrarily, and label the itemsets $A_1, A_2, \ldots, A_m$ according to this ordering. For a given $1 \leq K \leq m$, we denote by $f_{\mathcal{D}}^{(K)}$ the frequency $f_{\mathcal{D}}(A_K)$ of the K-th most frequent itemset A_K, and define the set of top-K FIs (with their respective frequencies) as

$$\mathsf{TOPK}(\mathcal{D}, \mathcal{I}, K) = \mathsf{FI}\left(\mathcal{D}, \mathcal{I}, f_{\mathcal{D}}^{(K)}\right). \tag{3.1}$$

One of the main uses of frequent itemsets is in the discovery of association rules. An *association rule* W is an expression "$A \Rightarrow B$" where A and B are itemsets such that $A \cap B = \emptyset$. The *support* $s_{\mathcal{D}}(W)$ (resp. frequency $f_{\mathcal{D}}(W)$) of the association rule W is the support (resp. frequency) of the itemset $A \cup B$. The *confidence* $c_{\mathcal{D}}(W)$ of W is the ratio $f_{\mathcal{D}}(A \cup B)/f_{\mathcal{D}}(A)$. Intuitively, an association rule "$A \Rightarrow B$" expresses, through its support and confidence, how likely it is for the itemset B to appear in the same transactions as itemset A. The confidence of the association rule can be interpreted the conditional probability of B being present in a transaction that contains A. Many other measures can be used to quantify the interestingness of an association rule [Tan et al., 2004]

[1] We assume $\mathcal{I} = \cup_{\tau \in \mathcal{D}} \tau$, i.e., all the elements of $\mathcal{I}$ appear in at least one transaction from $\mathcal{D}$.

(see also Sect. 3.4.4).

Definition 5. Given a dataset $\mathcal{D}$ with transactions built on a ground set $\mathcal{I}$, and given a minimum frequency threshold θ and a minimum confidence threshold γ, the *ARs task with respect to θ and γ* is to identify the set

$$\mathsf{AR}(\mathcal{D},\mathcal{I},\theta,\gamma) = \{(W, f_{\mathcal{D}}(W), c_{\mathcal{D}}(W)) \mid \text{association rule } W, f_{\mathcal{D}}(W) \geq \theta, c_{\mathcal{D}}(W) \geq \gamma\}\,.$$

We say that an itemset A (resp. an association rule W) is in $\mathsf{FI}(\mathcal{D},\mathcal{I},\theta)$ or in $\mathsf{TOPK}(\mathcal{D},\mathcal{I},K)$ (resp. in $\mathsf{AR}(\mathcal{D},\mathcal{I},\theta,\gamma)$) when there A (resp. W) is part of a pair in $\mathsf{FI}(\mathcal{D},\mathcal{I},\theta)$ or $\mathsf{TOPK}(\mathcal{D},\mathcal{I},K)$, (resp. a triplet $\mathsf{AR}(\mathcal{D},\mathcal{I},\theta,\gamma)$).

We are interested in extracting absolute and relative approximations of the sets $\mathsf{FI}(\mathcal{D},\mathcal{I},\theta)$, $\mathsf{TOPK}(\mathcal{D},\mathcal{I},K)$ and $\mathsf{AR}(\mathcal{D},\mathcal{I},\theta,\gamma)$.

Definition 6. Given a parameter ε_{abs} (resp. ε_{rel}), an *absolute ε_{abs}-close approximation* (resp. a *relative ε_{rel}-close approximation*) of $\mathsf{FI}(\mathcal{D},\mathcal{I},\theta)$ is a set $\mathcal{C} = \{(A, f_A) \ : \ A \subseteq \mathcal{I}, f_A \in [0,1]\}$ of pairs (A, f_A) where f_A approximates $f_{\mathcal{D}}(A)$. $\mathcal{C}$ is such that:

1. $\mathcal{C}$ contains all itemsets appearing in $\mathsf{FI}(\mathcal{D},\mathcal{I},\theta)$;
2. $\mathcal{C}$ contains no itemset A with frequency $f_{\mathcal{D}}(A) < \theta - \varepsilon_{\text{abs}}$ (resp. $f_{\mathcal{D}}(A) < (1-\varepsilon_{\text{rel}})\theta$);
3. For every pair $(A, f_A) \in \mathcal{C}$, it holds $|f_{\mathcal{D}}(A) - f_A| \leq \varepsilon_{\text{abs}}$ (resp. $|f_{\mathcal{D}}(A) - f_A| \leq \varepsilon_{\text{rel}} f_{\mathcal{D}}(A)$).

As an example, consider a dataset $\mathcal{D}$ where transactions have all length one and are built on the ground set $\mathcal{I} = \{a, b, c, d\}$. Suppose that $f_{\mathcal{D}}(a) = 0.4$, $f_{\mathcal{D}}(b) = 0.3$, $f_{\mathcal{D}}(c) = 0.2$, and $f_{\mathcal{D}}(d) = 0.1$ (clearly there are no other itemsets). If we set $\theta = 0.22$ and $\varepsilon = 0.05$, an absolute ε-close approximation C of $\mathsf{FI}(\mathcal{D},\mathcal{I},\theta)$ *must* contain two pairs (a, f_a) and (b, f_b) as $a, b \in \mathsf{FI}(Ds,\mathcal{I},\theta)$. At the same time, C *might* contain a pair (c, f_c), because $f_{\mathcal{D}}(c) > \theta - \varepsilon$. On the other hand C *must not* contain a pair (d, f_d) because $f_{\mathcal{D}}(d) < \theta - \varepsilon$. The values f_a, f_b, and eventually f_c must be not more than ε far from $f_{\mathcal{D}}(a)$, $f_{\mathcal{D}}(b)$, and $f_{\mathcal{D}}(c)$, respectively.

The above definition extends easily to the case of top-K frequent itemsets mining using the equivalence

$$\mathsf{TOPK}(\mathcal{D},\mathcal{I},K) = \mathsf{FI}\left(\mathcal{D},\mathcal{I},f_{\mathcal{D}}^{(K)}\right):$$

an absolute (resp. relative) ε-close approximation to $\mathsf{FI}\left(\mathcal{D},\mathcal{I},f_{\mathcal{D}}^{(K)}\right)$ is an absolute (resp. relative) ε-close approximation to $\mathsf{TOPK}(\mathcal{D},\mathcal{I},K)$.

For the case of association rules, we have the following definition.

Definition 7. Given a parameter ε_{abs} (resp. ε_{rel}), an *absolute ε_{abs}-close approximation* (resp. a *relative ε_{rel}-close approximation*) of $\mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$ is a set

$$\mathcal{C} = \{(W, f_W, c_W) \ : \ \text{association rule } W, f_W \in [0,1], c_W \in [0,1]\}$$

of triplets (W, f_W, c_W) where f_W and c_W approximate $f_{\mathcal{D}}(W)$ and $c_{\mathcal{D}}(W)$ respectively. $\mathcal{C}$ is such that:

1. $\mathcal{C}$ contains all association rules appearing in $\mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$;
2. $\mathcal{C}$ contains no association rule W with frequency $f_{\mathcal{D}}(W) < \theta - \varepsilon_{\text{abs}}$ (resp. $f_{\mathcal{D}}(W) < (1-\varepsilon_{\text{rel}})\theta$);
3. For every triplet $(W, f_W, c_W) \in \mathcal{C}$, it holds $|f_{\mathcal{D}}(W) - f_W| \leq \varepsilon_{\text{abs}}$ (resp. $|f_{\mathcal{D}}(W) - f_W| \leq \varepsilon_{\text{rel}}\theta$).
4. $\mathcal{C}$ contains no association rule W with confidence $c_{\mathcal{D}}(W) < \gamma - \varepsilon_{\text{abs}}$ (resp. $c_{\mathcal{D}}(W) < (1-\varepsilon_{\text{rel}})\gamma$);
5. For every triplet $(W, f_W, c_W) \in \mathcal{C}$, it holds $|c_{\mathcal{D}}(W) - c_W| \leq \varepsilon_{\text{abs}}$ (resp. $|c_{\mathcal{D}}(W) - c_W| \leq \varepsilon_{\text{rel}} c_{\mathcal{D}}(W)$).

Note that the definition of relative ε-close approximation to $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ (resp. to $\mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$) is more stringent than the definition of ε-close solution to frequent itemset mining (resp. association rule mining) in [Chakaravarthy et al., 2009, Sect. 3]. Specifically, we require an approximation of the frequencies (and confidences) in addition to the approximation of the collection of itemsets or association rules (Property 3 in Def. 6 and properties 3 and 5 in Def. 7).

3.3 The dataset's range space and its VC-dimension

Our next step is to define a range space of the dataset and the itemsets. We will use this space together with Theorem 1 to compute the bounds to sample sizes sufficient to compute approximate solutions for the various tasks of market basket analysis.

Definition 8. Let $\mathcal{D}$ be a dataset of transactions that are subsets of a ground set $\mathcal{I}$. We define $S = (D, \mathcal{R})$ to be a range space associated with $\mathcal{D}$ and $\mathcal{I}$ such that:

1. $D = \mathcal{D}$ is the set of transactions in the dataset.
2. $\mathcal{R} = \{T_{\mathcal{D}}(A) \mid A \subseteq \mathcal{I}, A \neq \emptyset\}$ is a family of sets of transactions such that for each non-empty itemset $A \subseteq \mathcal{I}$, the set $T_{\mathcal{D}}(A) = \{\tau \in \mathcal{D} \mid A \subseteq \tau\}$ of all transactions containing A is an element of $\mathcal{R}$.

It is easy to see that in practice the collection $\mathcal{R}$ of ranges contains all and only the sets $T_{\mathcal{D}}(A)$ where A is a *closed itemset*, i.e., a set such that for each non-empty $B \subseteq A$ we have $T_{\mathcal{D}}(B) = T_{\mathcal{D}}(A)$

and for any $C \supset A$, $T_{\mathcal{D}}(C) \subsetneq T_{\mathcal{D}}(A)$. Closed itemsets are used to summarize the collection of frequent itemsets [Calders et al., 2006].

The VC-Dimension of this range space is the maximum size of a set of transactions that can be shattered by the support sets of the itemsets, as expressed by the following theorem and the following corollary.

Theorem 4. *Let $\mathcal{D}$ be a dataset and let $S = (\mathcal{D}, \mathcal{R})$ be the associated range space. Let $v \in \mathbb{N}$. Then $\mathsf{VC}(S) \geq v$ if and only if there exists a set $\mathcal{A} \subseteq \mathcal{D}$ of v transactions from $\mathcal{D}$ such that for each subset $\mathcal{B} \subseteq \mathcal{A}$, there exists an itemset $I_{\mathcal{B}}$ such that the support set of $I_{\mathcal{B}}$ in $\mathcal{A}$ is exactly $\mathcal{B}$, that is $T_{\mathcal{A}}(I_{\mathcal{B}}) = \mathcal{B}$.*

Proof. "$\Leftarrow$". From the definition of $I_{\mathcal{B}}$, we have that $T_{\mathcal{D}}(I_{\mathcal{B}}) \cap \mathcal{A} = \mathcal{B}$. By definition of $P_{\mathcal{R}}(\mathcal{A})$ this means that $\mathcal{B} \in P_{\mathcal{R}}(\mathcal{A})$, for any subset $\mathcal{B}$ of $\mathcal{A}$. Then $P_{\mathcal{R}}(\mathcal{A}) = 2^{\mathcal{A}}$, which implies $\mathsf{VC}(S) \geq v$.

"$\Rightarrow$". Let $\mathsf{VC}(S) \geq v$. Then by the definition of VC-Dimension there is a set $\mathcal{A} \subseteq \mathcal{D}$ of v transactions from $\mathcal{D}$ such that $P_{\mathcal{R}}(A) = 2^{\mathcal{A}}$. By definition of $P_{\mathcal{R}}(\mathcal{A})$, this means that for each subset $\mathcal{B} \subseteq \mathcal{A}$ there exists an itemset $I_{\mathcal{B}}$ such that $T_{\mathcal{D}}(I_{\mathcal{B}}) \cap \mathcal{A} = \mathcal{B}$. We want to show that no transaction $\rho \in \mathcal{A} \setminus \mathcal{B}$ contains $I_{\mathcal{B}}$. Assume now by contradiction that there is a transaction $\rho^* \in \mathcal{A} \setminus \mathcal{B}$ containing $I_{\mathcal{B}}$. Then $\rho^* \in T_{\mathcal{D}}(I_{\mathcal{B}})$ and, given that $\rho^* \in \mathcal{A}$, we have $\rho^* \in T_{\mathcal{D}}(I_{\mathcal{B}}) \cap \mathcal{A}$. But by construction, we have that $T_{\mathcal{D}}(I_{\mathcal{B}}) \cap \mathcal{A} = \mathcal{B}$ and $\rho^* \notin \mathcal{B}$ because $\rho^* \in \mathcal{A} \setminus \mathcal{B}$. Then we have a contradiction, and there can not be such a transaction ρ^*. □

Corollary 1. *Let $\mathcal{D}$ be a dataset and $S = (\mathcal{D}, \mathcal{R})$ be the corresponding range space. Then the VC-Dimension $\mathsf{VC}(S)$ of S is the maximum integer v such that there is a set $\mathcal{A} \subseteq \mathcal{D}$ of v transactions from $\mathcal{D}$ such that for each subset $\mathcal{B} \subseteq \mathcal{A}$ of $\mathcal{A}$, there exists an itemset $I_{\mathcal{B}}$ such that the support of $I_{\mathcal{B}}$ in $\mathcal{A}$ is exactly $\mathcal{B}$, that is $T_{\mathcal{A}}(I_{\mathcal{B}}) = \mathcal{B}$.*

For example, consider the dataset $\mathcal{D} = \{\{a, b, c, d\}, \{a, b\}, \{a, c\}, \{d\}\}$ of four transactions built on the set of items $\mathcal{I} = \{a, b, c, d\}$. It is easy to see that the set of transactions $\mathcal{A} = \{\{a, b\}, \{a, c\}\}$ can be shattered: $\mathcal{A} = \mathcal{A} \cap T_{\mathcal{D}}(\{a\})$, $\{\{a, b\}\} = \mathcal{A} \cap T_{\mathcal{D}}(\{a, b\})$, $\{\{a, c\}\} = \mathcal{A} \cap T_{\mathcal{D}}(\{a, c\})$, $\emptyset = \mathcal{A} \cap T_{\mathcal{D}}(\{d\})$. It should be clear that there is no set of three transactions in $\mathcal{D}$ that can be shattered, so the VC-dimension of the range space associated to $\mathcal{D}$ is exactly two.

Computing the exact VC-dimension of the range space associated to a dataset is extremely expensive from a computational point of view. This does not come as a surprise, as it is known that computing the VC-dimension of a range space (X, R) can take time $O(|R||X|^{\log |R|})$ [Linial et al., 1991, Thm. 4.1]. It is instead possible to give an upper bound to the VC-dimension and a procedure to efficiently compute the bound.

We now define a characteristic quantity of the dataset, called the *d-index* and show that it is a tight bound to the VC-dimension of the range space associated to the dataset, then present an

algorithm to efficiently compute an upper bound to the d-index with a single linear scan of the dataset.

Definition 9. Let $\mathcal{D}$ be a dataset. The *d-index* of $\mathcal{D}$ is the maximum integer d such that $\mathcal{D}$ contains at least d different transactions of length at least d such that no one of them is a subset of another, that is, the transactions form an *anti-chain*.

Consider now the dataset $\mathcal{D} = \{\{a, b, c, d\}, \{a, b, d\}, \{a, c\}, \{d\}\}$ of four transactions built on the set of items $\mathcal{I} = \{a, b, c, d\}$. The d-index of $\mathcal{D}$ is 2, as the transactions $\{a, b, d\}$ and $\{a, c\}$ form an anti-chain. Note that the anti-chain determining the d-index is not necessarily the largest anti-chain that can be built on the transactions of $\mathcal{D}$. For example, if $\mathcal{D} = \{\{a, b, c, d\}, \{a, b\}, \{a, c\}, \{a\}, \{b\}, \{c\}, \{d\}\}$, the largest anti-chain would be $\{\{a\}, \{b\}, \{c\}, \{d\}\}$, but the anti-chain determining the d-index of the dataset would be $\{\{a, b\}, \{a, c\}, \{d\}\}$.

Intuitively, the reason for considering an anti-chain of transactions is that, if τ is a transaction that is a subset of another transaction τ', ranges containing τ' necessarily also contain τ (the opposite is not necessarily true), so it would be impossible to shatter a set containing both transactions.

It is easy to see that the d-index of a dataset built on a set of items $\mathcal{I}$ is at most equal to the length of the longest transaction in the dataset and in any case no greater than $|\mathcal{I}| - 1$.

The d-index is an upper bound to the VC-dimension of a dataset.

Theorem 5. *Let $\mathcal{D}$ be a dataset with d-index d. Then the range space $S = (\mathcal{D}, \mathcal{R})$ corresponding to $\mathcal{D}$ has VC-dimension at most d.*

Proof. Let $\ell > d$ and assume that S has VC-dimension ℓ. From Def. 1 there is a set $\mathcal{K}$ of ℓ transactions of $\mathcal{D}$ that is shattered by $\mathcal{R}$. Clearly, $\mathcal{K}$ cannot contain any transaction equal to $\mathcal{I}$, because such transaction would appear in all ranges of $\mathcal{R}$ and so it would not be possible to shatter $\mathcal{K}$. At the same time, for any two transactions τ, τ' in $\mathcal{K}$ we must have neither $\tau \subseteq \tau'$ nor $\tau' \subseteq \tau$, otherwise the shorter transaction of the two would appear in all ranges where the longer one appears, and so it would not be possible to shatter $\mathcal{K}$. Then $\mathcal{K}$ must be an anti-chain. From this and from the definitions of d and ℓ, $\mathcal{K}$ must contain a transaction τ such that $|\tau| \leq d$. The transaction τ is a member of $2^{\ell-1}$ subsets of $\mathcal{K}$. We denote these subsets of $\mathcal{K}$ containing τ as $\mathcal{A}_i$, $1 \leq i \leq 2^{\ell-1}$, labeling them in an arbitrary order. Since $\mathcal{K}$ is shattered (i.e., $P_{\mathcal{R}}(\mathcal{K}) = 2^{\mathcal{K}}$), we have

$$\mathcal{A}_i \in P_{\mathcal{R}}(\mathcal{K}), 1 \leq i \leq 2^{\ell-1}.$$

From the above and the definition of $P_{\mathcal{R}}(\mathcal{K})$, it follows that for each set of transactions $\mathcal{A}_i$ there must be a non-empty itemset B_i such that

$$T_{\mathcal{D}}(B_i) \cap \mathcal{K} = \mathcal{A}_i \in P_{\mathcal{R}}(\mathcal{K}). \tag{3.2}$$

Since the $\mathcal{A}_i$ are all different from each other, this means that the $T_{\mathcal{D}}(B_i)$ are all different from each other, which in turn requires that the B_i be all different from each other, for $1 \leq i \leq 2^{\ell-1}$.

Since $\tau \in \mathcal{A}_i$ and $\tau \in \mathcal{K}$ by construction, it follows from (3.2) that

$$\tau \in T_{\mathcal{D}}\left(B_i\right), 1 \leq i \leq 2^{\ell-1}.$$

From the above and the definition of $T_{\mathcal{D}}(B_i)$, we get that all the itemsets $B_i, 1 \leq i \leq 2^{\ell-1}$ appear in the transaction τ. But $|\tau| \leq d < \ell$, therefore τ can only contain at most $2^d - 1 < 2^{\ell-1}$ non-empty itemsets, while there are $2^{\ell-1}$ different itemsets B_i.

This is a contradiction, therefore our assumption is false and $\mathcal{K}$ cannot be shattered by $\mathcal{R}$, which implies that $\mathsf{VC}(S) \leq d$. □

This bound is strict, i.e., there are indeed datasets with VC-dimension exactly d, as formalized by the following Theorem.

Theorem 6. *There exists a dataset $\mathcal{D}$ with d-index d and such the corresponding range space has VC-dimension exactly d.*

Proof. For $d = 1$, $\mathcal{D}$ can be any dataset with at least two different transactions $\tau = \{a\}$ and $\tau' = \{b\}$ of length 1. The set $\{\tau\} \subseteq \mathcal{D}$ is shattered because $T_{\mathcal{D}}(\{a\}) \cap \{\tau\} = \{\tau\}$ and $T_{\mathcal{D}}(\{b\}) \cap \{\tau\} = \emptyset$.

Without loss of generality, let the ground set $\mathcal{I}$ be $\mathbb{N}$. For a fixed $d > 1$, let $\tau_i = \{0, 1, 2, \ldots, i-1, i+1, \ldots, d\}$ $1 \leq i \leq d$, and consider the set of d transactions $\mathcal{K} = \{\tau_i, 1 \leq i \leq d\}$. Note that $|\tau_i| = d$ and $|\mathcal{K}| = d$ and for no pair of transactions τ_i, τ_j with $i \neq j$ we have either $\tau_i \subseteq \tau_j$ nor $\tau_j \subseteq \tau_i$.

$\mathcal{D}$ is a dataset containing $\mathcal{K}$ and any number of arbitrary transactions from $2^{\mathcal{I}}$ of length at most d. Let $S = (\mathcal{D}, \mathcal{R})$ be the range space corresponding to $\mathcal{D}$. We now show that $\mathcal{K} \subseteq \mathcal{D}$ is shattered by ranges from $\mathcal{R}$, which implies $\mathsf{VC}(S) \geq d$.

For each $\mathcal{A} \in 2^{\mathcal{K}} \setminus \{\mathcal{K}, \emptyset\}$, let $Y_{\mathcal{A}}$ be the itemset

$$Y_{\mathcal{A}} = \{1, \ldots, d\} \setminus \{i \ : \ \tau_i \in \mathcal{A}\}.$$

Let $Y_{\mathcal{K}} = \{0\}$ and let $Y_{\emptyset} = \{d+1\}$. By construction we have

$$T_{\mathcal{K}}(Y_{\mathcal{A}}) = \mathcal{A}, \forall \mathcal{A} \subseteq \mathcal{K}$$

i.e., the itemset $Y_{\mathcal{A}}$ appears in all transactions in $\mathcal{A} \subseteq \mathcal{K}$ but not in any transaction from $\mathcal{K} \setminus \mathcal{A}$, for all $\mathcal{A} \in 2^{\mathcal{K}}$. This means that

$$T_{\mathcal{D}}(Y_{\mathcal{A}}) \cap \mathcal{K} = T_{\mathcal{K}}(Y_{\mathcal{A}}) = \mathcal{A}, \forall \mathcal{A} \subseteq \mathcal{K}.$$

Since for all $\mathcal{A} \subseteq \mathcal{K}, T_{\mathcal{D}}(Y_{\mathcal{A}}) \in R$ by construction, the above implies that

$$\mathcal{A} \in P_{\mathcal{R}}(\mathcal{K}), \forall \mathcal{A} \subseteq \mathcal{K}$$

This means that $\mathcal{K}$ is shattered by $\mathcal{R}$, hence $\mathsf{VC}(S) \geq d$. From this and Thm. 5, we can conclude that $\mathsf{VC}(S) = d$. □

Consider again the dataset $\mathcal{D} = \{\{a, b, c, d\}, \{a, b\}, \{a, c\}, \{d\}\}$ of four transactions built on the set of items $\mathcal{I} = \{a, b, c, d\}$. We argued before that the VC-dimension of the range space associated to this dataset is exactly two, and it is easy to see that the d-index of $\mathcal{D}$ is also two.

3.3.1 Computing the d-index of a dataset

The d-index of a dataset $\mathcal{D}$ exactly can be obtained in polynomial time by computing, for each length ℓ, the size w_ℓ of the largest anti-chain that can be built using the transactions of length at least ℓ from $\mathcal{D}$. If $w \geq \ell$, then the d-index is at least ℓ. The maximum ℓ for which $w_\ell \geq \ell$ is the d-index of $\mathcal{D}$. The size of the largest anti-chain that can be built on the elements of a set can be computed by solving a maximum matching problem on a bipartite graph that has two nodes for each element of the set [Ford and Fulkerson, 1962]. Computing the maximum matching can be done in polynomial time [Hopcroft and Karp, 1973].

In practice, this approach can be quite slow as it requires, for each value taken by ℓ, a scan of the dataset to create the set of transactions of length at least ℓ, and to solve a maximum matching problem. Hence, we now present an algorithm to efficiently compute an upper bound q to the d-index with a single linear scan of the dataset and with $O(q)$ memory.

It is easy to see that the d-index of a dataset $\mathcal{D}$ is upper bounded by the maximum integer q such that $\mathcal{D}$ contains at least q different (that is not containing the same items) transactions of length at least q and less than $|\mathcal{I}|$. This upper bound, which we call *d-bound*, ignores the constraint that the transactions that concur to the computation of the d-index must form an anti-chain. We can compute the d-bound in a greedy fashion by scanning the dataset once and keeping in memory the maximum integer q such that we saw at least q transactions of length q until this point of the scanning. We also keep in memory the q longest different transactions, to avoid counting transactions that are equal to ones we have already seen because, as we already argued, a set containing identical transactions can not be shattered and copies of a transaction should not be included in the computation of the d-index, so it is not useful to include them in the computation of the d-bound. The pseudocode

Algorithm 1: Compute the d-bound, an upper bound to the d-index of a dataset

Input : a dataset $\mathcal{D}$
Output: the d-bound q, an upper bound to the d-index of $\mathcal{D}$

```
1  τ ← getNextTransaction(D)
2  T ← {τ}
3  q ← 1
4  while scanIsNotComplete() do
5  |  τ ← getNextTransaction(D)
6  |  if |τ| > q and τ ≠ I and ¬∃a ∈ T such that τ = a then
7  |  |  R ← T ∪ {τ}
8  |  |  q ← max integer such that R contains at least q transactions of length at least q
9  |  |  T ← set of the q longest transactions from R (ties broken arbitrarily)
10 |  end
11 end
12 return q
```

for computing the d-bound in the way we just described is presented in Algorithm 1. The function `getNextTransaction` returns one transaction at a time from the dataset. Note though that this does not imply that, in a disk-based system, the algorithm needs a random read for each transaction. If the dataset is stored in a block-based fashion, one can read one block at a time and scan all transactions in that block, given that the order in which the transactions are scanned is not relevant for the correctness of the algorithm. This means that in the worst-case the algorithm performs a random read per block. The following lemma deals with the correctness of the algorithm.

Lemma 3. *The algorithm presented in Algorithm 1 computes the maximum integer q such that $\mathcal{D}$ contains at least q different transactions of length at least q and less than $|\mathcal{I}|$.*

Proof. The algorithm maintains the following invariant after each update of $\mathcal{T}$: the set $\mathcal{T}$ contains the ℓ longest (ties broken arbitrarily) different transactions of length at least ℓ, where ℓ is the maximum integer r for which, up until to this point of the scan, the algorithm saw at least r different transactions of length at least r. It should be clear that if the invariant holds after the scanning is completed, the thesis follows because the return value q is exactly the size $|\mathcal{T}| = \ell$ after the last transaction has been read and processed.

It is easy to see that this invariant is true after the first transaction has been scanned. Suppose now that the invariant is true at the beginning of the $n+1$-th iteration of the while loop, for any n, $0 \leq n \leq |\mathcal{D}| - 1$. We want to show that it will still be true at the end of the $n+1$-th iteration. Let τ be the transaction examined at the $n+1$-th iteration of the loop. If $\tau = |\mathcal{I}|$, the invariant is still true at the end of the $n+1$-th iteration, as ℓ does not change and neither does $\mathcal{T}$ because the test of the condition on line 6 of Algorithm 1 fails. The same holds if $|\tau| < \ell$. Consider now the case $|\tau| > \ell$. If $\mathcal{T}$ contained, at the beginning of the $n+1$-th iteration, one transaction equal to τ, then clearly ℓ would not change and neither does $\mathcal{T}$, so the invariant is still true at the end of the

$n+1$-th iteration. Suppose now that $|\tau| > \ell$ and that $\mathcal{T}$ did not contain any transaction equal to τ. Let ℓ_i be, for $i = 1, \ldots, |\mathcal{D}| - 1$, the value of ℓ at the end of the i-th iteration, and let $\ell_0 = 1$. If $\mathcal{T}$ contained, at the beginning of the $n+1$-th iteration, zero transactions of length ℓ_n, then necessarily it contained ℓ_n transactions of length greater than ℓ_n, by our assumption that the invariant was true at the end of the n-th iteration. Since $|\tau| > \ell_n$, it follows that $\mathcal{R} = \mathcal{T} \cup \{\tau\}$ contains $\ell_n + 1$ transactions of size at least $\ell_n + 1$, hence the algorithm at the end of the $n+1$-th iteration has seen $\ell_n + 1$ transactions of length at least $\ell_n + 1$, so $\ell = \ell_{n+1} = \ell_n + 1$. This implies that at the end of iteration $n+1$ the set $\mathcal{T}$ must have size $\ell_{n+1} = \ell_n + 1$, i.e., must contain one transaction more than at the beginning of the $n+1$-th iteration. This is indeed the case as the value q computed on line 8 of Algorithm 1 is exactly $|\mathcal{R}| = \ell_n + 1$ because of what we just argued about $\mathcal{R}$, and therefore $\mathcal{T}$ is exactly $\mathcal{R}$ at the end of the $n+1$-th iteration and contains the $\ell = \ell_{n+1}$ longest different transactions of length at least ℓ, which is exactly what is expressed by the invariant. If instead $\mathcal{T}$ contained, at the beginning of the $n+1$-th iteration, one or more transactions of length ℓ_n, then $\mathcal{T}$ contains at most $\ell_n - 1$ transactions of length greater than ℓ_n, and $\mathcal{R}$ contains at most ℓ_n transactions of length at least $\ell_n + 1$, hence $q = \ell_n$. This also means that the algorithm has seen, before the beginning of the $n+1$-th iteration, at most $\ell_n - 1$ different transactions strictly longer than ℓ_n. Hence, after seeing τ, the algorithm has seen at most ℓ_n transactions of length at least $\ell_n + 1$, so at the end of the $n+1$-th iteration we will have $\ell = \ell_{n+1} = \ell_n$. This means that the size of $\mathcal{T}$ at the end of the $n+1$-th iteration is the same as it was at the beginning of the same iteration. This is indeed the case because of what we argued about q. At the end of the $n+1$-th iteration, $\mathcal{T}$ contains 1) all transactions of length greater than ℓ_n that it already contained at the end of the n-th iteration, and 2) the transaction τ, and 3) all but one the transactions of length ℓ_n that it contained at the end of the n-th iteration. Hence the invariant is true at the end of the $n+1$-th iteration because ℓ did not change and we replaced in $\mathcal{T}$ a transaction of length ℓ_n with a longer transaction, that is, τ. Consider now the case of $|\tau| = \ell$. Clearly if there is a transaction in $\mathcal{T}$ that is equal to τ, the invariant is still true at the end of the $n+1$-th iteration, as ℓ does not change and $\mathcal{T}$ stays the same. If $\mathcal{T}$ did not contain, at the beginning of the $n+1$-th iteration, any transaction equal to τ, then also in this case ℓ would not change, that is $\ell = \ell_{n+1} = \ell_n$, because by definition of ℓ the algorithm already saw at least ℓ different transactions of length at least ℓ. This implies that $\mathcal{T}$ must have, at the end of the $n+1$-th iteration, the same size that it had at the beginning of the $n+1$-th iteration. This is indeed the case because the set $\mathcal{R}$ contains $\ell + 1$ different transactions of size at least ℓ, but there is no value $b > \ell$ for which $\mathcal{R}$ contains b transactions of length at least b, because of what we argued about ℓ, hence $|\mathcal{T}| = q = \ell$. At the end of the $n+1$-th iteration the set $\mathcal{T}$ contains 1) all the transactions of length greater than ℓ that it contained at the beginning of the $n+1$-th iteration, and 2) enough transactions of length ℓ to make $|\mathcal{T}| = \ell$. This means that $\mathcal{T}$ can contain, at the end of the $n+1$-th iteration, exactly the same set of transactions that it contained at the beginning $n+1$-th iteration and since, as we argued, ℓ does not change, then the invariant is still true at the end of the $n+1$-th iteration. This completes our proof that the invariant still holds at the end of the $n+1$ iteration for any n, and therefore holds at the end of the algorithm, proving the thesis. $\square$

The fact that the computation of the d-bound can be easily performed with a single linear scan of the dataset in an online greedy fashion makes it extremely practical also for updating the bound as new transactions are added to the dataset.

3.3.2 Speeding up the VC-dimension approximation task

We showed that the computation of the d-index or of the d-bound can be efficiently performed, and especially the latter only requires a single linear scan of the dataset, in a block-by-block fashion if the dataset is stored on disk. In some settings this may still be an expensive operation. We now present two ways to reduce the cost of this operation.

Empirical VC-dimension. It is possible to create a random sample $\mathcal{S}$ of the dataset $\mathcal{D}$ of the desired size $|\mathcal{S}|$, compute the d-index or the d-bound *on the sample*, which is less expensive than computing it on the whole dataset and, for the d-bound, can be done while creating $\mathcal{S}$, and finally, after fixing δ, use Thm. 2 to compute the ε for which $\mathcal{S}$ is an ε-approximation. Thus, we have a faster method for estimating the VC-dimension that, as we will show in the Sect. 3.4, can be used to extract an absolute ε-close approximation to the collection of (top-K) FIs and ARs.

Estimating the d-index from the transaction length distribution. When a Bayesian approach is justified, one views the dataset $\mathcal{D}$ as a sample of n transactions generated by a random process with some known (or assumed) priors. A number of mixture models have been proposed in the literature for modeling dataset generation, the most commonly used is the Dirichlet Process Mixture model [He and Shapiro, 2012]. In general, we assume that the generating process $\pi_{\bar{\alpha}}$ belongs to a known parametrized family of distributions $\Pi(\alpha)$ where α represents the parameters of the distribution. Deriving the parameter $\bar{\alpha}$ corresponding to the distribution of transaction lengths according to which the dataset $\mathcal{D}$ was generated can be done by sampling transactions from $\mathcal{D}$ and using techniques for parameter estimation for a distribution from $\Pi(\alpha)$ [Hastie et al., 2009; Lehmann and Casella, 1998]. Once the parameter $\bar{\alpha}$ is (probabilistically) known, an upper bound b to the d-index d can be easily derived (probabilistically). Estimating the parameter $\bar{\alpha}$ through sampling may take less time than performing a scan of the entire dataset to compute the d-bound, especially when the dataset is very large, a fast sequential sampling algorithm like Vitter's Method D [Vitter, 1987] is used, and the estimation procedure is fast.

3.3.3 Connection with monotone monomials

There is an interesting connection between itemsets built on a ground set $\mathcal{I}$ and the class of *monotone monomials on* $|\mathcal{I}|$ *literals*[2]. A *monotone* monomial is a conjunction of literals with no negations. The class MONOTONE-MONOMIALS$_{|\mathcal{I}|}$ is the class of all monotone monomials on $|\mathcal{I}|$ Boolean variables, including the constant functions $\mathbf{0}$ and $\mathbf{1}$. The VC-Dimension of the range space

$$(\{0,1\}^{|\mathcal{I}|}, \text{MONOTONE-MONOMIALS}_{|\mathcal{I}|})$$

is exactly $|\mathcal{I}|$ [Natschläger and Schmitt, 1996, Coroll. 3]. It is easy to see that it is always possible to build a bijective map between the itemsets in $2^{\mathcal{I}}$ and the elements of MONOTONE-MONOMIALS$_{|\mathcal{I}|}$ and that transactions built on the items in $\mathcal{I}$ correspond to points of $\{0,1\}^{|\mathcal{I}|}$. This implies that a dataset $\mathcal{D}$ can be seen as a sample from $\{0,1\}^{|\mathcal{I}|}$.

Solving the problems we are interested in by using the VC-Dimension $|\mathcal{I}|$ of monotone-monomials as an upper bound to the VC-dimension of the itemsets would have resulted in a much larger sample size than what is sufficient, given that $|\mathcal{I}|$ can be much larger than the d-index of a dataset. Instead, the VC-dimension of the range space $(\mathcal{D}, R)$ associated to a dataset $\mathcal{D}$ is equivalent to the VC-dimension of the range space $(\mathcal{D}, \text{MONOTONE-MONOMIALS}_{|\mathcal{I}|})$, which is the *empirical VC-Dimension* of the range space $(\{0,1\}^{|\mathcal{I}|}, \text{MONOTONE-MONOMIALS}_{|\mathcal{I}|})$ measured on $\mathcal{D}$. Our results, therefore, also show a tight bound to the empirical VC-Dimension of the class of monotone monomials on $|\mathcal{I}|$ variables.

3.4 Mining (top-K) Frequent Itemsets and Association Rules

We apply the VC-dimension results to constructing efficient sampling algorithms with performance guarantees for approximating the collections of FIs, top-K FIs and ARs.

3.4.1 Mining Frequent Itemsets

We construct bounds for the sample size needed to obtain relative/absolute ε-close approximations to the collection of FIs. The algorithms to compute the approximations use a standard exact FIs mining algorithm on the sample, with an appropriately adjusted minimum frequency threshold, as formalized in the following lemma.

[2]This connection was suggested to us by Luc De Raadt, to whom we are grateful.

Lemma 4. *Let $\mathcal{D}$ be a dataset with transactions built on a ground set $\mathcal{I}$, and let v be the VC-dimension of the range space associated to $\mathcal{D}$. Let $0 < \varepsilon, \delta < 1$. Let $\mathcal{S}$ be a random sample of $\mathcal{D}$ with size*

$$|\mathcal{S}| = \min\left\{|\mathcal{D}|, \frac{4c}{\varepsilon^2}\left(v + \log\frac{1}{\delta}\right)\right\},$$

for some absolute constant c. Then $\mathsf{FI}(\mathcal{S}, \mathcal{I}, \theta - \varepsilon/2)$ is an absolute ε-close approximation to $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ with probability at least $1 - \delta$.

Proof. Suppose that $\mathcal{S}$ is a $\varepsilon/2$-approximation of the range space $(\mathcal{D}, \mathcal{R})$ corresponding to $\mathcal{D}$. From Thm. 1 we know that this happens with probability at least $1 - \delta$. This means that for all $X \subseteq \mathcal{I}$, $f_{\mathcal{S}}(X) \in [f_{\mathcal{D}}(X) - \varepsilon/2, f_{\mathcal{D}}(X) + \varepsilon/2]$. This holds in particular for the itemsets in $\mathcal{C} = \mathsf{FI}(\mathcal{S}, \mathcal{I}, \theta - \varepsilon/2)$, which therefore satisfies Property 3 from Def. 6. It also means that for all $X \in \mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$, $f_{\mathcal{S}}(X) \geq \theta - \varepsilon/2$, so $\mathcal{C}$ also guarantees Property 1 from Def. 6. Let now $Y \subseteq \mathcal{I}$ be such that $f_{\mathcal{D}}(Y) < \theta - \varepsilon$. Then, for the properties of $\mathcal{S}$, $f_{\mathcal{S}}(Y) < \theta - \varepsilon/2$, i.e., $Y \notin \mathcal{C}$, which allows us to conclude that $\mathcal{C}$ also has Property 2 from Def. 6. □

We stress again that here and in the following theorems, the constant c is absolute and does not depend on $\mathcal{D}$ or on d, ε, or δ.

One very interesting consequence of this result is that we do not need to know in advance the minimum frequency threshold θ in order to build the sample: the properties of the ε-approximation allow to use the same sample for any threshold and for different thresholds, i.e., the sample does not need to be rebuilt if we want to mine it with a threshold θ first and with another threshold θ' later.

It is important to note that the VC-dimension associated to a dataset, and therefore the sample size from (2.2) needed to probabilistically obtain an ε-approximation, is independent from the size (number of transactions) in $\mathcal{D}$ and also of the size of $\mathsf{FI}(\mathcal{S}, \mathcal{I}, \theta)$. It is also always smaller or at most as large as the d-index d, which is always less or equal to the length of the longest transaction in the dataset, which in turn is less or equal to the number of different items $|\mathcal{I}|$.

To obtain a relative ε-close approximation, we need to add a dependency on θ as shown in the following Lemma.

Lemma 5. *Let $\mathcal{D}$, v, ε, and δ as in Lemma 4. Let $\mathcal{S}$ be a random sample of $\mathcal{D}$ with size*

$$|\mathcal{S}| = \min\left\{|\mathcal{D}|, \frac{4(2+\varepsilon)c}{\varepsilon^2\theta(2-\varepsilon)}\left(v\log\frac{2+\varepsilon}{\theta(2-\varepsilon)} + \log\frac{1}{\delta}\right)\right\},$$

for some absolute absolute constant c. Then $\mathsf{FI}(\mathcal{S}, \mathcal{I}, (1 - \varepsilon/2)\theta)$ is a relative ε-close approximation to $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ with probability at least $1 - \delta$.

Proof. Let $p = \theta(2-\varepsilon)/(2+\varepsilon)$. From Thm. 1, the sample $\mathcal{S}$ is a relative $(p, \varepsilon/2)$-approximation of the range space associated to $\mathcal{D}$ with probability at least $1-\delta$. For any itemset X in $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$, we have

$f_{\mathcal{D}}(X) \geq \theta > p$, so $f_{\mathcal{S}}(X) \geq (1-\varepsilon/2)f_{\mathcal{D}}(X) \geq (1-\varepsilon/2)\theta$, which implies $X \in \mathsf{FI}(\mathcal{S},\mathcal{I},(1-\varepsilon/2)\theta))$, so Property 1 from Def. 6 holds. Let now X be an itemsets with $f_{\mathcal{D}}(X) < (1-\varepsilon)\theta$. From our choice of p, we always have $p > (1-\varepsilon)\theta$, so $f_{\mathcal{S}}(X) \leq p(1+\varepsilon/2) < \theta(1-\varepsilon/2)$. This means $X \notin \mathsf{FI}(\mathcal{S},\mathcal{I},(1-\varepsilon/2)\theta))$, as requested by Property 2 from Def. 6. Since $(1-\varepsilon/2)\theta = p(1+\varepsilon/2)$, it follows that only itemsets X with $f_{\mathcal{D}}(X) \geq p$ can be in $\mathsf{FI}(\mathcal{S},\mathcal{I},(1-\varepsilon/2)\theta))$. For these itemsets it holds $|f_{\mathcal{S}}(X) - f_{\mathcal{D}}(X)| \leq f_{\mathcal{D}}(X)\varepsilon/2$, as requested by Property 3 from Def.6. □

3.4.2 Mining top-K Frequent Itemsets

Given the equivalence $\mathsf{TOPK}(\mathcal{D},\mathcal{I},K) = \mathsf{FI}(\mathcal{D},\mathcal{I},f_{\mathcal{D}}^{(K)})$, we could use the above FIs sampling algorithms if we had a good approximation of $f_{\mathcal{D}}^{(K)}$, the threshold frequency of the top-K FIs.

For the absolute ε-close approximation we first execute a standard top-K FIs mining algorithm on the sample to estimate $f_{\mathcal{D}}^{(K)}$ and then run a standard FIs mining algorithm on the same sample using a minimum frequency threshold depending on our estimate of $f_{\mathcal{S}}^{(K)}$. Lemma 6 formalizes this intuition.

Lemma 6. *Let $\mathcal{D}$, v, ε, and δ be as in Lemma 4. Let K be a positive integer. Let $\mathcal{S}$ be a random sample of $\mathcal{D}$ with size*

$$|\mathcal{S}| = \min\left\{|\mathcal{D}|, \frac{16c}{\varepsilon^2}\left(v + \log\frac{1}{\delta}\right)\right\}$$

, for some absolute constant c, then $\mathsf{FI}(\mathcal{S},\mathcal{I},f_{\mathcal{S}}^{(K)} - \varepsilon/2)$ is an absolute ε-close approximation to $\mathsf{TOPK}(\mathcal{D},\mathcal{I},K)$ with probability at least $1-\delta$.

Proof. Suppose that $\mathcal{S}$ is a $\varepsilon/4$-approximation of the range space (X,R) corresponding to $\mathcal{D}$. From Thm. 1 we know that this happens with probability at least $1-\delta$. This means that for all $Y \subseteq \mathcal{I}$, $f_{\mathcal{S}}(Y) \in [f_{\mathcal{D}}(Y) - \varepsilon/4, f_{\mathcal{D}}(Y) + \varepsilon/4]$. Consider now $f_{\mathcal{S}}^{(K)}$, the frequency of the K-th most frequent itemset in the sample. Clearly, $f_{\mathcal{S}}^{(K)} \geq f_{\mathcal{D}}^{(K)} - \varepsilon/4$, because there are at least K itemsets (for example any subset of size K of $\mathsf{TOPK}(\mathcal{D},\mathcal{I},K)$) with frequency in the sample at least $f_{\mathcal{D}}^{(K)} - \varepsilon/4$. On the other hand $f_{\mathcal{S}}^{(K)} \leq f_{\mathcal{D}}^{(K)} + \varepsilon/4$, because there cannot be K itemsets with a frequency in the sample greater than $f_{\mathcal{D}}^{(K)} + \varepsilon/4$: only itemsets with frequency in the dataset strictly greater than $f_{\mathcal{D}}^{(K)}$ can have a frequency in the sample greater than $f_{\mathcal{D}}^{(K)} + \varepsilon/4$, and there are at most $K-1$ such itemsets. Let now $\eta = f_{\mathcal{S}}^{(K)} - \varepsilon/2$, and consider $\mathsf{FI}(\mathcal{S},\mathcal{I},\eta)$. We have $\eta \leq f_{\mathcal{D}}^{(K)} - \varepsilon/4$, so for the properties of $\mathcal{S}$, $\mathsf{TOPK}(\mathcal{D},\mathcal{I},K) = \mathsf{FI}(\mathcal{D},\mathcal{I},f_{\mathcal{D}}^{(K)}) \subseteq \mathsf{FI}(\mathcal{S},\mathcal{I},\eta)$, which then guarantees Property 1 from Def. 6. On the other hand, let Y be an itemset such that $f_{\mathcal{D}}(Y) < f_{\mathcal{D}}^{(K)} - \varepsilon$. Then $f_{\mathcal{S}}(Y) < f_{\mathcal{D}}^{(K)} - 3\varepsilon/4 \leq \eta$, so $Y \notin \mathsf{FI}(\mathcal{S},\mathcal{I},\eta)$, corresponding to Property 2 from Def. 6. Property 3 from Def. 6 follows from the properties of $\mathcal{S}$. □

Note that as in the case of the sample size required for an absolute ε-close approximation to $\mathsf{FI}(\mathcal{D},\mathcal{I},\theta)$, we do not need to know K in advance to compute the sample size for obtaining an

absolute ε-close approximation to $\mathsf{TOPK}(\mathcal{D}, \mathcal{I}, K)$.

Two different samples are needed for computing a relative ε-close approximation to $\mathsf{TOPK}(\mathcal{D}, \mathcal{I}, K)$, the first one to compute a lower bound to $f_{\mathcal{D}}^{(K)}$, the second to extract the approximation. Details for this case are presented in Lemma 7.

Lemma 7. *Let $\mathcal{D}$, v, ε, and δ be as in Lemma 4. Let K be a positive integer. Let δ_1, δ_2 be two reals such that $(1-\delta_1)(1-\delta_2) \geq (1-\delta)$. Let $\mathcal{S}_1$ be a random sample of $\mathcal{D}$ with some size*

$$|\mathcal{S}_1| = \frac{\phi c}{\varepsilon^2}\left(v + \log\frac{1}{\delta_1}\right)$$

for some $\phi > 2\sqrt{2}/\varepsilon$ and some absolute constant c. If $f_{\mathcal{S}_1}^{(K)} \geq (2\sqrt{2})/(\varepsilon\phi)$, then let $p = (2-\varepsilon)\theta/(2+\varepsilon)$ and let $\mathcal{S}_2$ be a random sample of $\mathcal{D}$ of size

$$|\mathcal{S}_2| = \min\left\{|\mathcal{D}|, \frac{4c}{\varepsilon^2 p}(v\log\frac{1}{p} + \log\frac{1}{\delta})\right\}$$

for some absolute constant c. Then $\mathsf{FI}(\mathcal{S}_2, \mathcal{I}, (1-\varepsilon/2)(f_{\mathcal{S}_1}^{(K)} - \varepsilon/\sqrt{2\phi}))$ is a relative ε-close approximation to $\mathsf{TOPK}(\mathcal{D}, \mathcal{I}, K)$ with probability at least $1-\delta$.

Proof. Assume that $\mathcal{S}_1$ is a $\varepsilon/\sqrt{2\phi}$-approximation for $\mathcal{D}$ and $\mathcal{S}_2$ is a relative $(p, \varepsilon/2)$-approximation for $\mathcal{D}$. The probability of these two events happening at the same time is at least $1-\delta$, from Thm. 1.

Following the steps of the proof of Lemma 6 we can easily get that, from the properties of $\mathcal{S}_1$,

$$f_{\mathcal{S}_1}^{(K)} - \frac{\varepsilon}{\sqrt{2\phi}} \leq f_{\mathcal{D}}^{(K)} \leq f_{\mathcal{S}_1}^{(K)} + \frac{\varepsilon}{\sqrt{2\phi}}. \tag{3.3}$$

Consider now an element $X \in \mathsf{TOPK}(\mathcal{D}, \mathcal{I}, K)$. We have by definition $f_{\mathcal{D}}(X) \geq f_{\mathcal{D}}^{(K)} > f_{\mathcal{S}_1}^{(K)} - \varepsilon/\sqrt{2\phi} \geq p$, and from the properties of $\mathcal{S}_2$, it follows that $f_{\mathcal{S}}(X) \geq (1-\varepsilon/2)f_{\mathcal{D}}(X) \geq (1-\varepsilon/2)(f_{\mathcal{S}_1}^{(K)} - \varepsilon/\sqrt{2\phi})$, which implies $X \in \mathsf{FI}(\mathcal{S}_2, \mathcal{I}, (1-\varepsilon/2)(f_{\mathcal{S}_1}^{(K)} - \varepsilon/\sqrt{2\phi}))$ and therefore Property 1 from Def. 6 holds for $FI(\mathcal{S}_2, \mathcal{I}, \eta)$.

Let now Y be an itemset such that $f_{\mathcal{D}}(Y) < (1-\varepsilon)f_{\mathcal{D}}^{(K)}$. From our choice of p we have that $f_{\mathcal{D}}(Y) < p$. Then $f_{\mathcal{S}_2}(Y) < (1+\varepsilon/2)p < (1-\varepsilon/2)(f_{\mathcal{S}_1}^{(K)} - \varepsilon/\sqrt{2\phi})$. Therefore, $Y \notin \mathsf{FI}(\mathcal{S}_2, \mathcal{I}, \eta)$ and Property 2 from Def. 6 is guaranteed.

Property 3 from Def. 6 follows from (3.3) and the properties of $\mathcal{S}_2$. □

3.4.3 Mining Association Rules

Our final theoretical contribution concerns the discovery of relative/absolute approximations to $\mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \eta)$ from a sample. Lemma 8 builds on a result from [Chakaravarthy et al., 2009, Sect. 5] and covers the *relative* case, while Lemma 9 deals with the *absolute* one.

Lemma 8. *Let $0 < \delta, \varepsilon, \theta, \gamma < 1$, $\phi = \max\{2+\varepsilon, 2-\varepsilon+2\sqrt{1-\varepsilon}\}$, $\eta = \varepsilon/\phi$, and $p = \theta(1-\eta)/(1+\eta)$. Let $\mathcal{D}$ be a dataset and v be the VC-dimension of the range space associated to $\mathcal{D}$. Let $\mathcal{S}$ be a random sample of $\mathcal{D}$ of size*

$$|\mathcal{S}| = \min\left\{|\mathcal{D}|, \frac{c}{\eta^2 p}\left(v\log\frac{1}{p} + \log\frac{1}{\delta}\right)\right\} \tag{3.4}$$

for some absolute constant c. Then $\mathsf{AR}(\mathcal{S}, \mathcal{I}, (1-\eta)\theta, \gamma(1-\eta)/(1+\eta))$ is a relative ε-close approximation to $\mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$ with probability at least $1-\delta$.

Proof. Suppose $\mathcal{S}$ is a relative (p, η)-approximation for the range space corresponding to $\mathcal{D}$. From Thm. 1 we know this happens with probability at least $1-\delta$.

Let $W \in \mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$ be the association rule "$A \Rightarrow B$", where A and B are itemsets. By definition $f_{\mathcal{D}}(W) = f_{\mathcal{D}}(A \cup B) \geq \theta > p$. From this and the properties of $\mathcal{S}$, we get

$$f_{\mathcal{S}}(W) = f_{\mathcal{S}}(A \cup B) \geq (1-\eta) f_{\mathcal{D}}(A \cup B) \geq (1-\eta)\theta.$$

Note that, from the fact that $f_{\mathcal{D}}(W) = f_{\mathcal{D}}(A \cup B) \geq \theta$, it follows that $f_{\mathcal{D}}(A), f_{\mathcal{D}}(B) \geq \theta > p$, for the anti-monotonicity Property of the frequency of itemsets.

By definition, $c_{\mathcal{D}}(W) = f_{\mathcal{D}}(W)/f_{\mathcal{D}}(A) \geq \gamma$. Then,

$$c_{\mathcal{S}}(W) = \frac{f_{\mathcal{S}}(W)}{f_{\mathcal{S}}(A)} \geq \frac{(1-\eta) f_{\mathcal{D}}(W)}{(1+\eta) f_{\mathcal{D}}(A)} \geq \frac{1-\eta}{1+\eta} \cdot \frac{f_{\mathcal{D}}(W)}{f_{\mathcal{D}}(A)} \geq \frac{1-\eta}{1+\eta}\gamma.$$

It follows that $W \in \mathsf{AR}(\mathcal{S}, \mathcal{I}, (1-\eta)\theta, \gamma(1-\eta)/(1+\eta))$, hence Property 1 from Def. 7 is satisfied.

Let now Z be the association rule "$C \Rightarrow D$", such that $f_{\mathcal{D}}(Z) = f_{\mathcal{D}}(C \cup D) < (1-\varepsilon)\theta$. But from our definitions of η and p, it follows that $f_{\mathcal{D}}(Z) < p < \theta$, hence $f_{\mathcal{S}}(Z) < (1+\eta)p < (1-\eta)\theta$, and therefore $Z \notin \mathsf{AR}(\mathcal{S}, \mathcal{I}, (1-\eta)\theta, \gamma(1-\eta)(1+\eta))$, as requested by Property 2 from Def. 7.

Consider now an association rule $Y =$ "$E \Rightarrow F$" such that $c_{\mathcal{D}}(Y) < (1-\varepsilon)\gamma$. Clearly, we are only concerned with Y such that $f_{\mathcal{D}}(Y) \geq p$, otherwise we just showed that Y can not be in $\mathsf{AR}(\mathcal{S}, \mathcal{I}, (1-\eta)\theta, \gamma(1-\eta)/(1+\eta))$. From this and the anti-monotonicity property, it follows that $f_{\mathcal{D}}(E), f_{\mathcal{D}}(F) \geq p$. Then,

$$c_{\mathcal{S}}(Y) = \frac{f_{\mathcal{S}}(Y)}{f_{\mathcal{S}}(E)} \leq \frac{(1+\eta) f_{\mathcal{D}}(Y)}{(1-\eta) f_{\mathcal{D}}(E)} < \frac{1+\eta}{1-\eta}(1-\varepsilon)\gamma < \frac{1-\eta}{1+\eta}\gamma,$$

where the last inequality follows from the fact that $(1-\eta)^2 > (1+\eta)(1-\varepsilon)$ for our choice of η. We can conclude that $Y \notin \mathsf{AR}(\mathcal{S}, \mathcal{I}, (1-\varepsilon)\theta, \gamma(1-\eta)/(1+\eta)\gamma)$ and therefore Property 4 from Def. 7 holds.

Properties 3 and 5 from Def. 7 follow from the above steps (i.e., what association rules can be in the approximations), from the definition of ϕ, and from the properties of $\mathcal{S}$. □

Lemma 9. *Let $\mathcal{D}$, v, θ, γ, ε, and δ be as in Lemma 8 and let $\varepsilon_{\text{rel}} = \varepsilon/\max\{\theta, \gamma\}$.*

Fix $\phi = \max\{2+\varepsilon, 2-\varepsilon_{\text{rel}}+2\sqrt{1-\varepsilon_{\text{rel}}}\}$, $\eta = \varepsilon_{\text{rel}}/\phi$, and $p = \theta(1-\eta)/(1+\eta)$. Let $\mathcal{S}$ be a

random sample of $\mathcal{D}$ of size

$$|\mathcal{S}| = \min\left\{|\mathcal{D}|, \frac{c}{\eta^2 p}\left(v \log\frac{1}{p} + \log\frac{1}{\delta}\right)\right\} \tag{3.5}$$

for some absolute constant c. Then $\mathsf{AR}(\mathcal{S}, \mathcal{I}, (1-\eta)\theta, \gamma(1-\eta)/(1+\eta))$ *is an absolute* ε*-close approximation to* $\mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$.

Proof. The thesis follows from Lemma 8 by setting ε there to ε_{rel}. □

Note that the sample size needed for absolute ε-close approximations to $\mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$ depends on θ and γ, which was not the case for absolute ε-close approximations to $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ and $\mathsf{TOPK}(\mathcal{D}, \mathcal{I}, K)$.

3.4.4 Other interestingness measures

Confidence is not the only measure for the interestingness of an association rule. Other measures include lift, IS (cosine), all-confidence, Jaccard index, leverage, conviction, and many more [Tan et al., 2004]. In this section we apply our general technique to obtain good approximations with respect to a number of these measures, while also showing the limitation of our technique with respect to other criteria.

We use the term absolute (or relative) ε-close approximation as defined in Def. 7, appropriately adapted to the relevant measure in place of the confidence. We also extend our notation and denote the collection of ARs with frequency at least θ and interestingness at least γ according to a measure w by $\mathsf{AR}_w(\mathcal{D}, \mathcal{I}, \theta, \gamma)$, that is, indicating the measure in the subscript of "AR".

The first two measures we deal with are *all-confidence* and *IS* (also known as *Cosine*). They are defined as follows:

$$\text{all-confidence:} \quad ac_{\mathcal{D}}(A \Rightarrow B) = \frac{f_{\mathcal{D}}(A \cup B)}{max_{a \in A \cup B} f_{\mathcal{D}}(A)}$$

$$\text{IS (Cosine):} \quad is_{\mathcal{D}}(A \Rightarrow B) = \frac{f_{\mathcal{D}}(A \cup B)}{\sqrt{f_{\mathcal{D}}(A) f_{\mathcal{D}}(B)}}$$

SInce the approximation errors in the enumerators and denominators of these measures are the same as in computing the confidence, we can follow exactly the same steps as in the proof of Lemmas 8 and 9 and obtain the same procedures, parameters, and sample sizes (3.4) and (3.5) to extract relative and absolute ε-close approximations to the collection of ARs according to these measures.

Lift. The *lift* of an association rule "$A \Rightarrow B$" is defined as

$$\ell_{\mathcal{D}}(A \Rightarrow B) = \frac{f_{\mathcal{D}}(A \cup B)}{f_{\mathcal{D}}(A) f_{\mathcal{D}}(B)} \ .$$

We have the following result about computing a relative ε-close approximation to the collection of ARs according to lift.

Lemma 10. *Let $\mathcal{D}$, v, θ, γ, ε, and δ be as in Lemma 8. There exists a value η such that, if we let $p = \theta(1-\eta)/(1+\eta)$, and let $\mathcal{S}$ be random sample of $\mathcal{D}$ of size*

$$|\mathcal{S}| = \min\left\{|\mathcal{D}|, \frac{c}{\eta^2 p}\left(v \log\frac{1}{p} + \log\frac{1}{\delta}\right)\right\}$$

for some absolute constant c, we have that $\mathsf{AR}_\ell(\mathcal{S}, \mathcal{I}, (1-\eta)\theta, \gamma(1-\eta)/(1+\eta))$ is a relative ε-close approximation to $\mathsf{AR}_\ell(\mathcal{D}, \mathcal{I}, \theta, \gamma)$.

Proof. In order for $\mathsf{AR}_\ell(\mathcal{S}, \mathcal{I}, (1-\eta)\theta, \gamma(1-\eta)/(1+\eta))$ to satisfy the properties of a relative ε-close approximation, η must be a solution to the following system of inequalities:

$$\begin{cases} (1-\varepsilon)(1+\eta)^3 < (1-\eta)^3 \\ \dfrac{1+\eta}{(1-\eta)^2} \leq 1+\varepsilon \\ \dfrac{1-\eta}{(1+\eta)^2} \geq 1-\varepsilon \\ 0 \leq \eta < 1 \end{cases}$$

The first inequality expresses the requirement of Property 4 from Def. 7. The second and the third inequality deal with Properties 1, 3, and 5. The last inequality limits the domain of η. Property 2 from Def. 7 would be enforced by the choice of p. It can be verified that this system admits solutions. Once the value of η has been determined, we can proceed as in the proof of Lemma 8 to prove that all properties from Def. 7 are satisfied. □

We can get a result about *absolute* ε-close approximation to $\mathsf{AR}_\ell(\mathcal{D}, \mathcal{I}, \theta, \gamma)$ by following the same derivation of Lemma 9.

Piatetsky-Shapiro measure (leverage). Another measure of interestingness is the *Piatetsky-Shapiro* measure (also known as *leverage*):

$$ps_{\mathcal{D}}(A \Rightarrow B) = f_{\mathcal{D}}(A \cup B) - f(A) f(B) \ .$$

We first prove that it is possible to obtain an *absolute* ε-close approximation to the collection of ARs according to this measure and then argue that our methods can not be used to obtain a *relative* ε-close approximation to such collection.

Lemma 11. *Let $\mathcal{D}$, v, θ, γ, ε, and δ be as in Lemma 8. Let $\mathcal{S}$ be a random sample of $\mathcal{D}$ of size*

$$|S| = \min\left\{|D|, \frac{64c}{\varepsilon^2}\left(v + \log\frac{1}{\delta}\right)\right\}$$

for some absolute constant c. Then $\mathsf{AR}_{ps}(\mathcal{S}, \mathcal{I}, \theta - \varepsilon/8, \gamma - \varepsilon/2)$ is an absolute ε-close approximation to $\mathsf{AR}_{ps}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$ with probability at least $1 - \delta$.

Proof. Assume that $\mathcal{S}$ is a $\varepsilon/8$-approximation for $\mathcal{D}$. From Thm. 1 we know this happens with probability at least $1-\delta$. This implies that for any itemeset $A \subseteq \mathcal{I}$, we have $|f_{\mathcal{D}}(A) - f_{\mathcal{S}}(A)| \leq \varepsilon/8$, which holds in particular for the association rules in $\mathsf{AR}_{ps}(\mathcal{S}, \mathcal{I}, \theta - \varepsilon/8, \gamma - \varepsilon/2)$, so Property 3 from Def. 7 is satisfied.

Consider now an association rule $W =$ "$A \Rightarrow B$". We have

$$\begin{aligned} ps_{\mathcal{S}}(W) &= f_{\mathcal{S}}(A \cup B) - f_{\mathcal{S}}(A) f_{\mathcal{S}}(B) \geq f_{\mathcal{D}}(A \cup B) - \frac{\varepsilon}{8} - \left(f_{\mathcal{D}}(A) + \frac{\varepsilon}{8}\right)\left(f_{\mathcal{D}}(B) + \frac{\varepsilon}{8}\right) \\ &\geq f_{\mathcal{D}}(A \cup B) - f_{\mathcal{D}}(A) f_{\mathcal{D}}(B) - \frac{\varepsilon}{8}\left(1 + f_{\mathcal{D}}(A) + f_{\mathcal{D}}(B) + \frac{\varepsilon}{8}\right) \\ &\leq ps_{\mathcal{D}}(W) - \frac{\varepsilon}{2} \ . \end{aligned} \tag{3.6}$$

We also have:

$$\begin{aligned} ps_{\mathcal{S}}(W) &= f_{\mathcal{S}}(A \cup B) - f_{\mathcal{S}}(A) f_{\mathcal{S}}(B) \leq f_{\mathcal{D}}(A \cup B) + \frac{\varepsilon}{8} - \left(f_{\mathcal{D}}(A) - \frac{\varepsilon}{8}\right)\left(f_{\mathcal{D}}(B) - \frac{\varepsilon}{8}\right) \\ &\leq f_{\mathcal{D}}(A \cup B) - f_{\mathcal{D}}(A) f_{\mathcal{D}}(B) + \frac{\varepsilon}{8}\left(1 + f_{\mathcal{D}}(A) + f_{\mathcal{D}}(B) - \frac{\varepsilon}{8}\right) \\ &\leq ps_{\mathcal{D}}(W) + \frac{\varepsilon}{2} \end{aligned} \tag{3.7}$$

From (3.6) and (3.7) we get that for any association rule W, we have $|ps_{\mathcal{D}}(W) - ps_{\mathcal{S}}(W)| < \varepsilon$, hence Property 5 from Def. 7 holds.

If $W \in \mathsf{AR}_{ps}(\mathcal{S}, \mathcal{I}, \theta, \gamma)$, (3.6) implies that $W \in \mathsf{AR}_{ps}(\mathcal{S}, \mathcal{I}, \theta - \varepsilon/2, \gamma - \varepsilon/2)$, therefore Property 1 from Def. 7 is satisfied.

Let now Z be an association rule with frequency $f_{\mathcal{D}}(Z) < \theta - \varepsilon$. From the property of $\mathcal{S}$, we have that $f_{\mathcal{S}}(Z) \leq f_{\mathcal{D}}(Z) + \varepsilon/8 < \theta - \varepsilon + \varepsilon/8 < \theta - \varepsilon/8$, so $Z \notin \mathsf{AR}_{ps}(\mathcal{S}, \mathcal{I}, \theta - \varepsilon/8, \gamma - \varepsilon/2)$, which proves Property 2 from Def. 7.

Consider now an association rule $Y =$ "$C \Rightarrow D$" with frequency $f_{\mathcal{D}}(Y) > \theta$ but leverage $ps_{\mathcal{D}}(Y) < \gamma - \varepsilon$ ($Y \notin \mathsf{AR}_{ps}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$). From (3.7) we get that $ps_{\mathcal{S}}(Y) < \gamma - \varepsilon/2$ which implies that $Y \notin \mathsf{AR}_{ps}(\mathcal{S}, \mathcal{I}, \theta - \varepsilon/8, \gamma - \varepsilon/2)$, hence proving Property 4 from Def. 7. This concludes our proof. □

We now argue that it is not possible, in general, to extend our methods to obtain a relative ε-close approximation to $\mathsf{AR}_{ps}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$. Suppose that there is a parameter λ for which, for any itemset

A, we can find a value $\tilde{f}(A)$ such that $(1-\lambda)f_{\mathcal{D}}(A) \leq \tilde{f}(A) \leq (1+\lambda)f_{\mathcal{D}}(A)$. Let $\widetilde{ps}(A \Rightarrow B) = \tilde{f}(A \cup B) - \tilde{f}(A)\tilde{f}(B)$. We would like to show that the values $\widetilde{ps}$ cannot be used to obtain a relative ε-close approximation to $\mathsf{AR}_{ps}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$ in general. $0 < \varepsilon, \theta, \gamma < 1$. Among the requirement for a relative ε-close approximation we have that for an association rule "$A \to B$" in the approximation, it must hold $\widetilde{ps}(A \Rightarrow B) \geq (1-\varepsilon)ps_{\mathcal{D}}(A \Rightarrow B)$. We now show that this is not true in general. We have the following:

$$\begin{aligned}
\widetilde{ps}(A \Rightarrow B) &\geq (1-\lambda)f_{\mathcal{D}}(A \cup B) - (1+\lambda)^2 f_{\mathcal{D}}(A)f_{\mathcal{D}}(B) \\
&\geq (1-\varepsilon)f_{\mathcal{D}}(A \cup B) - (1-\varepsilon)f_{\mathcal{D}}(A)f_{\mathcal{D}}(B) \iff \\
&(\varepsilon-\lambda)f_{\mathcal{D}}(A \cup B) - (\varepsilon + 2\lambda + \lambda^2)f_{\mathcal{D}}(A)f_{\mathcal{D}}(B) \geq 0
\end{aligned}$$

Clearly, the inequality on the last line may not be true in general. This means that we can not, in general, obtain a relative ε-close approximation to $\mathsf{AR}_{ps}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$ by approximating the frequencies of all itemsets, no matter how good these would be.

Other measures. For other measures it may not be possible or straightforward to analytically derive procedures and sample sizes sufficient to extract good approximations of the collection of ARs according to these measures. Nevertheless most of them express the interestingness of an association rule as a function of the frequencies of the itemsets involved in the rule. Because of this, in practice, high quality approximation of the frequencies of all itemsets should be sufficient to obtain good approximation of the interestedness of a rule, and therefore, good approximation of the collection of ARs.

3.4.5 Closed Frequent Itemsets

A Closed Frequent Itemset (CFI) is a FI A whose subsets have all the same frequency $f\_Ds(A)$ of A. The collection of CFIs is a lossless compressed representation of the FIs [Calders et al., 2006]. The collection of CFIs is quite sensitive to sampling, as shown by the following example. Consider the dataset

$$\mathcal{D} = \{\{a, b, c\}, \{a\}, \{b\}, \{c\}\}.$$

Suppose that $\theta = 0.5$. Then $\mathsf{FI}(\mathcal{D},\mathcal{I},\theta) = \{\{a\},\{b\},\{c\}\}$, and this is also the collection of CFIs. Consider the sample $\mathcal{S} = \{\{a,b,c\},\{b\}\}$ of $\mathcal{D}$. We have that

$$\mathsf{FI}(\mathcal{S},\mathcal{I},\theta') = \{\{a\},\{b\},\{c\},\{a,b\},\{a,c\},\{b,c\},\{a,b,c\}\}$$

for any $\theta' \leq \theta$. But the collection of CFIs is $\{\{b\},\{a,b,c\}\}$, and it is not a superset of the original collection. Thus, in a sample, a superset of an original CFI may become closed instead of the original one. Therefore, given an absolute ε-close approximation F to $\mathsf{FI}(\mathcal{D},\mathcal{I},\theta)$ (analogously for a relative approximation), one could obtain a superset of the original collection of CFIs by considering, for each CFI $B \in F$, the set of subsets of B whose frequency in F is less than 2ε far from that of B. As was the case for FIs, a single scan of the dataset is then sufficient to filter out spurious candidates that are not CFIs from the so-obtained collection.

3.4.6 Discussion

In the previous sections we presented the bounds to the sample sizes as function of the VC-Dimension v of the range space associated to the dataset. As we argued in Sect. 3.3, computing the VC-dimension exactly is not a viable option. We therefore introduced the d-index d and the d-bound q as upper bounds to the VC-dimension that are efficient to compute, as described in Sect. 3.3.1. In practice one would use d or q, rather than v, to obtain the needed sample sizes.

Chakaravarthy et al. [2009] presented bounds to the sample sizes that depend on the length Δ of the longest transaction. It should be clear that $v \leq d \leq q \leq \Delta$, with the first inequality being strict in the worst case (Thm. 6). In real datasets, we have that $v \leq d \leq q \ll \Delta$: a single very long transaction has minimal impact on the VC-dimension or its upper bounds. One can envision cases where an anomalous transaction contains most items from $\mathcal{I}$, while all other transactions have constant length. This would drive up the sample size from [Chakaravarthy et al., 2009], while the bounds we present would not be impacted by this anomality.

Moreover, in practice one could expect v to be much smaller than the d-index d. This is due to the fact that the d-index is really a worst case bound which should only occur in artificial datasets, as it should be evident from the proof of Thm. 6. It would be very interesting to investigate better methods to estimate the actual VC-dimension of the range space associated to a dataset, rather than upper-bound it with d or q, as this could lead to much smaller sample sizes. We discussed in Sect. 2.3 how the problem of estimating the VC-dimension of learning machines is a fundamental problem in

learning, given that analytical computation of the exact value is usually impossible, as it is in our case. The procedure proposed by Vapnik et al. [1994] and Shao et al. [2000] procedure, although applicable to our case under mild conditions, is not very practical. It is very highly time consuming, as it requires the creation and analysis of multiple artificial datasets starting from the original one. Developing efficient ways to estimate the VC-dimension of a range space is an interesting research problem.

We conclude this discussion noting that all the bounds we presented have a dependency on $1/\varepsilon^2$. This is due to the use of tail bounds dependent on this quantity in the proof of the bound (2.2) to the sample size needed to obtain an ε-approximation. Given that the bound (2.2) is in general tight up to a constant [Li et al., 2001], there seems to be little room for improvement of the bounds we presented as function of ε.

3.5 Experimental evaluation

In this section we present an extensive experimental evaluation of our methods to extract approximations of $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$, $\mathsf{TOPK}(\mathcal{D}, \mathcal{I}, K)$, and $\mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$.

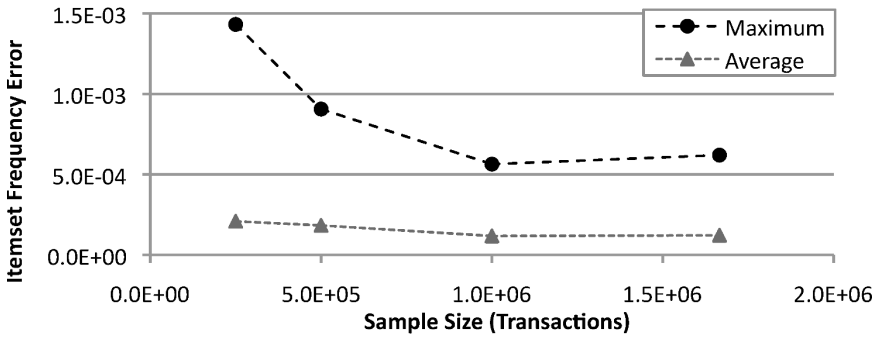
Our first goal is to evaluate the *quality* of the approximations obtained using our methods, by comparing the experimental results to the analytical bounds. We also evaluate how strict the bounds are by testing whether the same quality of results can be achieved at sample sizes smaller than those computed by the theoretical analysis. We then show that our methods can significantly speed-up the mining process, fulfilling the motivating promises of the use of sampling in the market basket analysis tasks. Lastly, we compare the sample sizes from our results to the best previous work [Chakaravarthy et al., 2009].

We tested our methods on both real and artificial datasets. The real datasets come from the FIMI'04 repository (`http://fimi.ua.ac.be/data/`). Since most of them have a moderately small size, we replicated their transactions a number of times, with the only effect of increasing the size of the dataset but no change in the distribution of the frequencies of the itemsets. The artificial datasets were built such that their corresponding range spaces have VC-dimension equal to the maximum transaction length, which is the maximum possible as shown in Thm. 5. To create these datasets, we followed the proof of Thm. 6 and used the generator included in ARtool (`http://www.cs.umb.edu/~laur/ARtool/`), which is similar to the one presented in [Agrawal and Srikant,

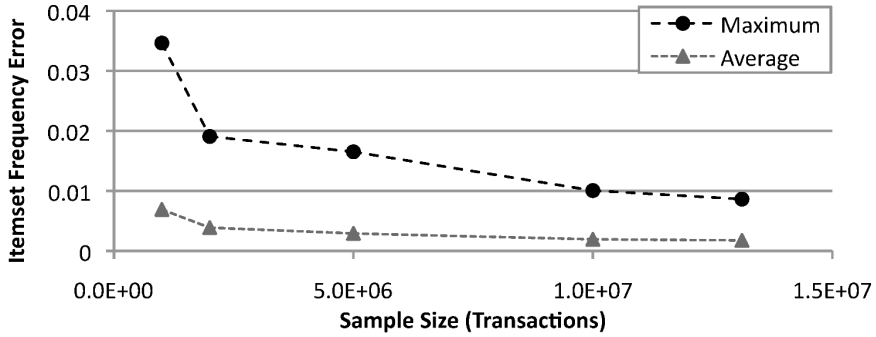
1994]. The artificial datasets had ten million transactions. We used the FP-Growth and Apriori implementations in ARtool to extract frequent itemsets and association rules. To compute the d-bound q, which is an upper bound to the d-index d, we used Algorithm 1. In all our experiments we fixed $\delta = 0.1$. In the experiments involving absolute (resp. relative) ε-close approximations we set $\varepsilon = 0.01$ (resp. $\varepsilon = 0.05$). The absolute constant c was fixed to 0.5 as estimated by [Löffler and Phillips, 2009]. This is reasonable because, again, c does not depend in any way from $\mathcal{D}$, ε, δ, the VC-dimension v of the range space, the d-index d or the d-bound q, or any characteristic of the collection of FIs or ARs. No upper bound is currently known for c' when computing the sizes for relative ε-approximations. We used the same value 0.5 for c' and found that it worked well in practice. For each dataset we selected a range of minimum frequency thresholds and a set of values for K when extracting the top-K frequent itemsets. For association rules discovery we set the minimum confidence threshold $\gamma \in \{0.5, 0.75, 0.9\}$. For each dataset and each combination of parameters we created random samples with size as computed by our theorems and with smaller sizes to evaluate the strictness of the bounds. We measured, for each set of parameters, the *absolute frequency error* and the *absolute confidence error*, defined as the error $|f_{\mathcal{D}}(X) - f_{\mathcal{S}}(X)|$ (resp. $|c_{\mathcal{D}}(Y) - c_{\mathcal{S}}(Y)|$) for an itemset X (resp. an association rule Y) in the approximate collection extracted from sample $\mathcal{S}$. When dealing with the problem of extracting *relative* ε-close approximations, we defined the *relative frequency error* to be the absolute frequency error divided by the real frequency of the itemset and analogously for the relative confidence error (dividing by the real confidence). In the figures we plot the maximum and the average for these quantities, taken over all itemsets or association rules in the output collection. In order to limit the influence of a single sample, we computed and plot in the figures the maximum (resp. the average) of these quantities in three runs of our methods on three different samples for each size.

The first important result of our experiments is that, for all problems (FIs, top-K FIs, ARs), for every combination of parameters, and for every run, the collection of itemsets or of association rules obtained using our methods always satisfied the requirements to be an absolute or relative ε-close approximation to the real collection. Thus in practice our methods indeed achieve or exceed the theoretical guarantees for approximations of the collections $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$, $\mathsf{TOPK}(\mathcal{D}, \mathcal{I}, \theta)$, and $\mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$. Given that the collections returned by our algorithms where always a superset of the collections of interest or, in other words, that the *recall* of the collections we returned was always 1.0, we measured the *precision* of the returned collection. In all but one case this statistic was at least

0.9 (out of a maximum of 1.0), suggesting relatively few false positives in the collections output. In the remaining case (extracting FIs from the dataset BMS-POS), the precision ranged between 0.59 to 0.8 (respectively for $\theta = 0.02$ and $\theta = 0.04$). The probability of including a FI or an AR which has frequency (or confidence, for ARs) less than θ (or γ) but does not violate the properties of a ε-close approximation, and is therefore an "acceptable" false positive, depends on the distribution of the real frequencies of the itemsets or ARs in the dataset around the frequency threshold θ (more precisely, below it, within ε or $\varepsilon\theta$): if many patterns have a real frequency in this interval, then it is highly probable that some of them will be included in the collections given in output, driving precision down. Clearly this probability depends on the number of patterns that have a real frequency close to θ. Given that usually the lower the frequency the higher the number of patterns with that frequency, this implies that our methods may include more "acceptable" false positives in the output at very low frequency thresholds. Once again, this depends on the distribution of the frequencies and does not violate the guarantees offered by our methods. It is possible to use the output of our algorithms as a set of *candidate patterns* which can be reduced to the real exact output (i.e., with no false positives) with a single scan of the dataset.



(a) Absolute Itemset Frequency Error, BMS-POS dataset, $d = 81$, $\theta = 0.02$, $\varepsilon = 0.01$, $\delta = 0.1$.



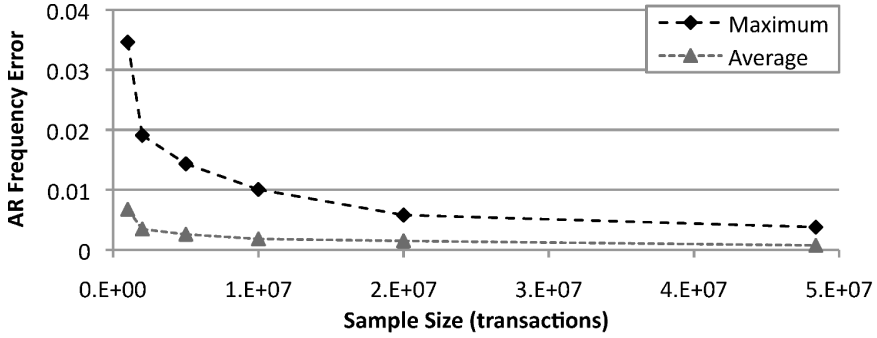
(b) Relative Itemset Frequency Error, artificial dataset, $v = 33$, $\theta = 0.01$, $\varepsilon = 0.05$, $\delta = 0.1$.

Figure 3.1: Absolute / Relative ε-close Approximation to $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$

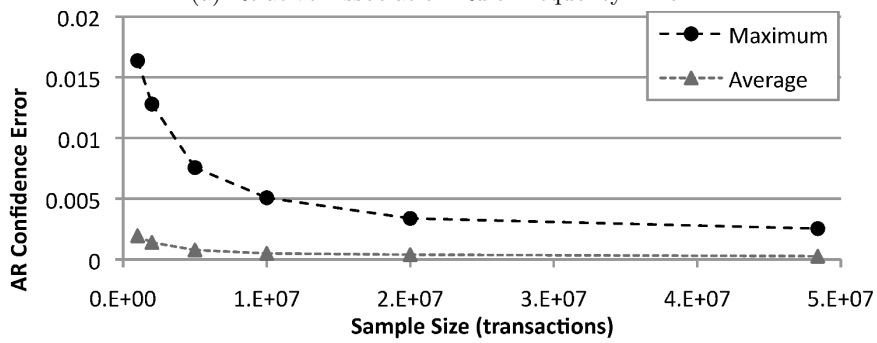
Evaluating the strictness of the bounds to the sample size was the second goal of our experiments. In Fig. 3.1a we show the behaviour of the maximum frequency error as function of the sample size in the itemsets obtained from samples using the method presented in Lemma 4 (i.e., we are looking for an *absolute* ε-close approximation to $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$). The rightmost plotted point corresponds to the sample size computed by the theoretical analysis. We are showing the results for the dataset BMS-POS replicated 40 times (d-index $d = 81$), mined with $\theta = 0.02$. It is clear from the picture that the guaranteed error bounds are achieved even at sample sizes smaller than what computed by the analysis and that the error at the sample size derived from the theory (rightmost plotted point for each line) is one to two orders of magnitude smaller than the maximum tolerable error $\varepsilon = 0.01$. This can be exmplained by the fact that the d-bound used to compute the sample size is in practice, as we argued in Sect. 3.4.6 a quite loose upper bound to the real VC-dimension. In Fig. 3.1b we report similar results for the problem of computing a *relative* ε-close approximation to $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ for an artificial dataset whose range space has VC-dimension v equal to the length of the longest transaction in the dataset, in this case 33. The dataset contained 100 million transactions. The sample size, given by Lemma 5, was computed using $\theta = 0.01$, $\varepsilon = 0.05$, and $\delta = 0.1$. The conclusions we can draw from the results for the behaviour of the relative frequency error are similar to those we got for the absolute case. For the case of absolute and relative ε-close approximation to $\mathsf{TOPK}(\mathcal{D}, \mathcal{I}, K)$, we observed results very similar to those obtained for $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$, as it was expected, given the closed connection between the two problems.

The results of the experiments to evaluate our method to extract a relative ε-close approximation to $\mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$ are presented in Fig. 3.2a and 3.2b. The same observations as before hold for the relative frequency error, while it is interesting to note that the relative confidence error is even smaller than the frequency error, most possibly because the confidence of an association rule is the ratio between the frequencies of two itemsets that appear in the same transactions and their sample frequencies will therefore have similar errors that cancel out when the ratio is computed. Similar conclusions can be made for the absolute ε-close approximation case.

From Fig. 3.1a, 3.1b, 3.2a, and 3.2b it is also possible to appreciate that, as the sample gets smaller, the maximum and the average errors in the frequency and confidence estimations increase. This suggests that using a fixed sampling rate or a fixed sample size can not guarantee good results for any ε: not only the estimation of the frequency and/or of the confidence would be quite off from the real value, but because of this, many itemsets that are frequent in the original dataset may be



(a) Relative Association Rule Frequency Error



(b) Relative Association Rule Confidence Error

Figure 3.2: Relative ε-close approximation to $\mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$, artificial dataset, $v = 33$, $\theta = 0.01$, $\gamma = 0.5$, $\varepsilon = 0.05$, $\delta = 0.1$.

missing from the output collection and many spurious (very infrequent) itemsets may be included in it.

The major motivating intuition for the use of sampling in market basket analysis tasks is that mining a sample of the dataset is faster than mining the entire dataset. Nevertheless, the mining time does not only depend on the number of transactions, but also on the number of frequent itemsets. Given that our methods suggest to mine the sample at a lowered minimum frequency threshold, this may cause an increase in running time that would make our method not useful in practice, because there may be many more frequent itemsets than at the original frequency threshold. We performed a number of experiments to evaluate whether this was the case and present the results in Fig. 3.3. We mined the artificial dataset introduced before for different values of θ, and created samples of size sufficient to obtain a relative ε-close approximation to $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$, for $\varepsilon = 0.05$ and $\delta = 0.1$. Figure 3.3 shows the time needed to mine the large dataset and the time needed to create and mine the samples. It is possible to appreciate that, even considering the sampling time, the speed up achieved by our method is around the order of magnitude (i.e. 10x speed improvement), proving the

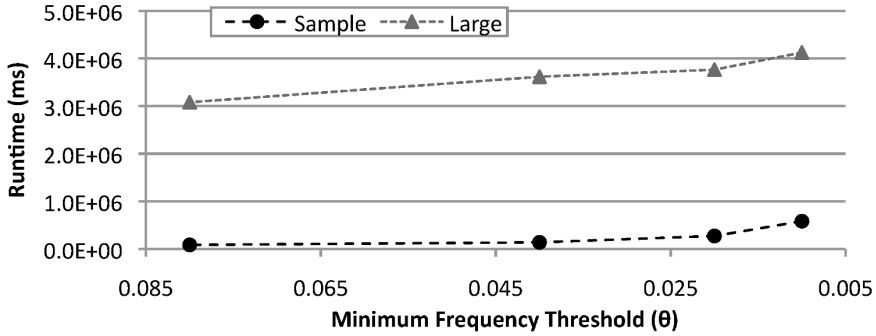
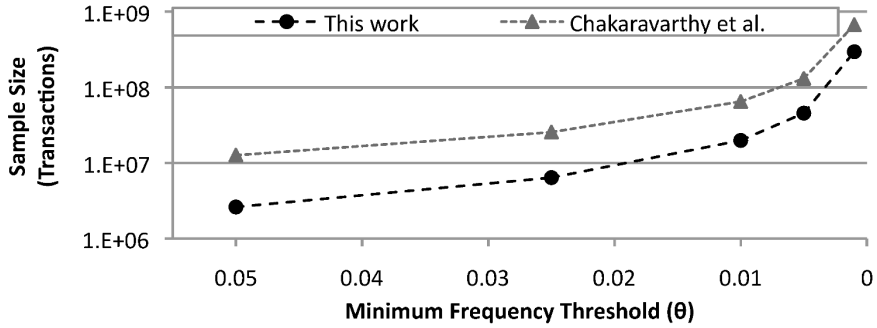


Figure 3.3: Runtime Comparison. The sample line includes the sampling time. Relative approximation to FIs, artificial dataset, $v = 33$, $\varepsilon = 0.05$, $\delta = 0.1$

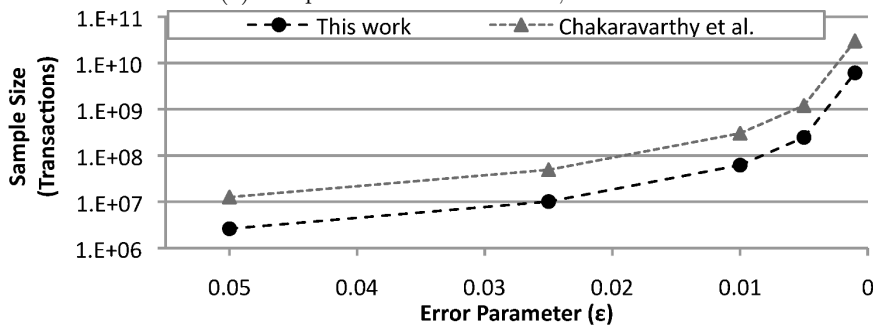
usefulness of sampling. Moreover, given that the sample size, and therefore the time needed to mine the sample, does not grow with the size of the dataset as long as the d-bound remains constant, that the d-index computation can be performed online, and that the time to create the sample can be made dependent only on the sample size using Vitter's Method D algorithm [Vitter, 1987], our method is very scalable as the dataset grows, and the speed up becomes even more relevant because the mining time for the large dataset would instead grow with the size of the dataset.

Comparing our results to previous work we note that the bounds generated by our technique are always linear in the VC-dimension v associated with the dataset. As reported in Table 3.1, the best previous work [Chakaravarthy et al., 2009] presented bounds that are linear in the maximum transaction length Δ for two of the six problems studied here. Figures 3.4a and 3.4b show a comparison of the actual sample sizes for relative ε-close approximations to $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ for as function of θ and ε. To compute the points for these figures, we set $\Delta = v = 50$, corresponding to the worst possible case for our method, i.e., when the VC-dimension of the range space associated to the dataset is exactly equal to the maximum transaction length. We also fixed $\delta = 0.05$ (the two methods behave equally as δ changes). For Fig. 3.4a, we fixed $\varepsilon = 0.05$, while for Fig. 3.4b we fixed $\theta = 0.05$. From the Figures we can appreciate that both bounds have similar, but not equal, dependencies on θ and ε. More precisely the bound we present is less dependent on ε and only slightly more dependent on θ. It also evident that the sample sizes given by the bound we present are always much smaller than those presented in [Chakaravarthy et al., 2009] (the vertical axis has logarithmic scale). In this comparison we used $\Delta = v$, but almost all real datasets we encountered have $v \ll \Delta$ as shown in Table 3.2 which would result in a larger gap between the sample sizes provided by the two methods. On the other hand, we should mention that the sample size given

by [Chakaravarthy et al., 2009] can be slightly optimized by using a stricter version of the Chernoff bound, but this does not change the fact that it depends on the maximum transaction length rather than on the VC-Dimension.



(a) Sample size as function of θ, $\varepsilon = 0.05$



(b) Sample size as function of ε, $\theta = 0.05$

Figure 3.4: Comparison of sample sizes for relative ε-close approximations to $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$. $\Delta = v = 50$, $\delta = 0.05$.

Table 3.2: Values for maximum transaction length Δ and d-bound q for real datasets

	accidents	BMS-POS	BMS-Webview-1	kosarak	pumsb*	retail	webdocs
Δ	51	164	267	2497	63	76	71472
q	46	81	57	443	59	58	2452

3.6 Conclusions

In this chapter we presented a novel technique to derive random sample sizes sufficient to easily extract high-quality approximations of the (top-K) frequent itemsets and of the collection of association rules. The sample size are linearly dependent on the VC-Dimension of the range space associated to the dataset, which is upper bounded by the maximum integer d such that there at

least d transactions of length at least d in the dataset. This bound is tight for a large family of datasets.

We used theoretical tools from statistical learning theory to develop a very practical solution to an important problem in computer science. The practicality of our method is demonstrated in the extensive experimental evaluation which confirming our theoretical analysis and suggests that in practice it is possible to achieve even better results than what the theory guarantees.

Samples of size as computed by our methods can be used to mine approximations of other collection of itemsets, provided that one correctly define the approximation taking into account the guarantees on the estimation of the frequency provided by the ε-approximation theorem. For example, one can can use techniques like those presented in [Mampaey et al., 2011] on a sample to obtain a small collection of patterns that describe the dataset as best as possible.

It may be possible to develop procedures that give a stricter upper bound to the VC-dimension for a given dataset, or that other measures of sample complexity like the triangular rank [Newman and Rabinovich, 2012], shatter coefficients, or Rademacher inequalities [Boucheron et al., 2005], can suggest smaller samples sizes.

In the next chapter we build on the results presented here to develop and analyze a fast and scalable algorithm for approximate FIs and ARs mining that leverages on the power of the MapReduce distributed/parallel framework to achieve very high accuracy and confidence while exploiting the available computational resources.

In Ch. 5 we flip the way we look at the itemsets mining problem and present a method to avoid the inclusion of *false positives* (itemsets that are frequent only by chance) in the output.

Chapter 4

PARMA: Mining Frequent Itemsets and Association Rules in MapReduce

We now extend the results presented in Chapter 3 and introduce PARMA, a randomized parallel algorithm for approximate frequent itemset mining, that makes the problem of Frequent Itemsets and Association Rules mining (FIM) embarassingly parallel, thus exhibiting near-linear speedup with the number of machines. PARMA combines random sampling and parallelization techniques in a novel fashion. It mines, in parallel, a set of small random samples and then filters and aggregates the collections of frequent itemsets or association rules obtained from each sample. Our work is orthogonal to other approaches, like PFP [Li et al., 2008], which focuses on parallelizing the mining phase in order to decrease the corresponding component of the cost. Due to the use of random sampling, the output of PARMA is an approximation of the collection of FIs or ARs in the dataset, but leveraging on the results presented in Chapter 3, PARMA offers tight probabilistic guarantees on the quality of the approximated collections returned in output. In particular it guarantees that the output is an ε-approximation of the real collection with probability at least $1 - \delta$, where ε and δ are parameters specified by the user (see Section 4.2 for formal definitions). PARMA is designed on MapReduce [Dean and Ghemawat, 2008], a novel parallel/distributed architecture that has raised significant interest in the research and industry communities. MapReduce is capable of handling

This chapter is an extended version of a work originally appeared in the proceedings of CIKM'12 as [Riondato et al., 2012].

very large datasets and efficiently executing parallel algorithms like PARMA.

To our knowledge PARMA is the first algorithm to exploit the combination of random sampling and parallelization for the task of Association Rules Mining.

A number of previous works explored either parallel algorithms [Buehrer et al., 2007; Cong et al., 2005; El-hajj and Zaïane, 2006; Fang et al., 2008; Liu et al., 2007; Özkural et al., 2011; Ruoming et al., 2005; Zaki, 1999] or random sampling (see Sect. 3.1) for the FIM task, but the two approaches have been seen somewhat orthogonal until today. In PARMA, the disadvantages of either approach are evened out by the advantages of the other. In the spirit of *moving computation to the data* to minimize communication, we avoid data replication, and preserve the advantages of parallelization by using of multiple independent small random samples of the dataset which are mined in parallel and have only their results aggregated. Similarly, we are not subject to the inherent trade-off between the size of the random sample and the accuracy of the approximation that can be obtained from it, as PARMA would only have to mine more samples of the same size in parallel to get higher quality approximations.

Although PARMA is not the first algorithm to use MapReduce to solve the FIM task, it differs from and enhances previous works [Cryans et al., 2010; Ghoting et al., 2011; Hammoud, 2011; Li et al., 2008; Li and Zhang, 2011; Yang et al., 2010; Zhou et al., 2010] in two crucial aspects. First, it significantly reduces the data that is replicated and transmitted in the *shuffle* phase of MapReduce. Second, PARMA is not limited to the extraction of Frequent Itemsets but can also directly compute the collection of Association Rules in MapReduce. In previous works, association rules had to be created sequentially after the Frequent Itemsets had been computed in MapReduce.

We conducted an extensive experimental evaluation to test the relative performance, scalability and accuracy of PARMA across a wide range of parameters and datasets. Our results suggest that PARMA can significantly outperform exact mining solutions, has near-linear speedup, and, as data and nodes are scaled together, is able to achieve near constant runtimes. Also, our accuracy evaluation shows that PARMA consistently computes approximated collections of higher quality than what can be analytically guaranteed.

In this chapter:

1. We present PARMA, the first randomized MapReduce algorithm for discovering approximate collections of frequent itemsets or association rules with near-linear speedup.

2. We provide analytical guarantees for the quality of the approximate results generated by the algorithm.

3. We demonstrate the effectiveness of PARMA on many datasets and compare the performance of our implementation to that of several exact FIM algorithms on MapReduce.

4.1 Previous work

We already discussed the relevant previous work about general FI mining and the use of sampling for this task in Sect. 3.1. We now focus on the contributions that are relevant for this chapter: the mining of FIs in a parallel or distributed setting, the MapReduce framework of computation, and its use to compute FIs and ARs.

The use of parallel and/or distributed algorithms for Association Rules Mining comes from the impossibility to handle very large datasets on a single machine. Early contributions in this area are presented in a survey by Zaki [1999]. In recent years, the focus shifted to exploit architecture advantages as much as possible, such as shared memory [Ruoming et al., 2005], cluster architecture [Buehrer et al., 2007] or the massive parallelism of GPUs [Fang et al., 2008]. The main goal is to avoid communication between nodes as much as possible and minimize the amount of data that are moved across the network [Cong et al., 2005; El-hajj and Zaïane, 2006; Liu et al., 2007; Özkural et al., 2011].

The MapReduce [Dean and Ghemawat, 2008] paradigm enjoys widespread success across both industry and academia. Research communities in many different fields uses this novel approach to distributed/parallel computation to develop algorithms to solve important problems in computer science [Chierichetti et al., 2010; Chu et al., 2007; Goodrich et al., 2011; Lin and Schatz, 2010; Pietracaprina et al., 2012]. Not only MapReduce can easily perform computation on very large datasets, but it is also extremely suited in executing embarassingly parallel algorithms which make a very limited use of communication. PARMA fits in this description so MapReduce is an appropriate choice for it.

A number of previous works [Cryans et al., 2010; Ghoting et al., 2011; Hammoud, 2011; Li et al., 2008; Li and Zhang, 2011; Yang et al., 2010; Zhou et al., 2010] looked at adapting APriori and FP-growth to the MapReduce setting. Somewhat naively, some authors [Cryans et al., 2010; Li and Zhang, 2011; Yang et al., 2010] suggest a distributed/parallel counting approach, i.e. to

compute the support of every itemset in the dataset in a single MapReduce round. This algorithm necessarily incurs in a huge data replication, given that an exponential number of messages are sent to the reducers, as each transaction contains a number of itemsets that is exponential in its length. A different adaptation of APriori to MapReduce is presented in [Hammoud, 2011, Chap.4]: similarly to the original formulation of APriori, at each round i, the support for itemsets of length i is computed, and those that are deemed frequent are then used to generate candidate frequent itemsets of length $i + 1$, although outside of the MapReduce environment. Apart from this, the major downsides of such approach are that some data replication still occurs, slowing down the shuffling phase, and that the algorithm does not complete until the longest frequent itemset is found. Given that length is not known in advance, the running time of the algorithm can not be computed in advance. Also the entire dataset needs to be scanned at each round, which can be very expensive, even if it is possible to keep additional data structures to speed up this phase.

An adaptation of FP-Growth to MapRreduce called PFP is presented by Li et al. [2008]. First, a parallel/distributed counting approach is used to compute the frequent items, which are then randomly partitioned into groups. Then, in a single MapReduce round the transactions in the dataset are used to generate group-dependent transactions. Each group is assigned to a reducer and the corresponding group-dependent transactions are sent to this reducer which then builds the local FP-tree and the conditional FP-trees recursively, computing the frequent patterns. The group-dependent transactions are such that the local FP-trees and the conditional FP-trees built by different reducers are independent. This algorithm suffers from a data replication problem: the number of group-dependent transactions generated for each single transaction is potentially equal to the number of groups. This means that the dataset may be replicated up to a number of times equal to the number of groups, resulting in a huge amount of data to be sent to the reducers and therefore in a slower synchronization/communication (*shuffle*) phase, which is usually the most expensive in a MapReduce algorithm. Another practical downside of PFP is that the time needed to mine the dependent FP-tree is not uniform across the groups. An empirical solution to this load balancing problem is presented by Zhou et al. [2010], although with no guarantees and by computing the groups outside the MapReduce environment. An implementation of the PFP algorithm as presented in [Li et al., 2008] is included in Apache Mahout (`http://mahout.apache.org`). Ghoting et al. [2011] present an high-level library to perform various machine learning and data mining tasks using MapReduce. They show how to implement the Frequent Itemset Mining task using their library.

The approach is very similar to that in [Li et al., 2008], and the same observations apply about the performances and downsides of this approach.

4.2 Preliminaries

In this chapter we show how to extract ε-close approximations of the collection of (top-k) Frequent Itemsets and Association Rules using MapReduce (see Sect. 3.2 for definitions about FIs and ARs). We slightly generalize the definitions of ε-close approximation of $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ (Def. 6) and of $\mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$ (Def. 6) as follows.

Definition 10. Given two parameters $\varepsilon_1, \varepsilon_2 \in (0, 1)$, an *absolute* (resp. *relative*) $(\varepsilon_1, \varepsilon_2)$-*close approximation* of $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ is a set $\mathcal{C} = \{(A, f_A, \mathcal{K}_A) \ : \ A \in 2^{\mathcal{I}}, f_A \in \mathcal{K}_A \subseteq [0, 1]\}$ of triplets $(A, f_A, \mathcal{K}_A)$ where f_A approximates $f_{\mathcal{D}}(A)$ and $\mathcal{K}_A$ is an interval containing f_A and $f_{\mathcal{D}}(A)$. $\mathcal{C}$ is such that:

1. $\mathcal{C}$ contains all itemsets appearing in $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$;
2. $\mathcal{C}$ contains no itemset A with frequency $f_{\mathcal{D}}(A) < \theta - \varepsilon_1$ (resp. $f_{\mathcal{D}}(A) \leq \theta(1 - \varepsilon_1)$;
3. For every triplet $(A, f_A, \mathcal{K}_A) \in \mathcal{C}$, it holds
   (a) $|f_{\mathcal{D}}(A) - f_A| \leq \varepsilon_2$ (resp. $|f_{\mathcal{D}}(A) - f_A| \leq \varepsilon_2 f_{\mathcal{D}}(A)$).
   (b) f_A and $f_{\mathcal{D}}(A)$ belong to $\mathcal{K}_A$.
   (c) $|\mathcal{K}_A| \leq 2\varepsilon_2$ (resp. $|\mathcal{K}_A| \leq 2\varepsilon_2 f_{\mathcal{D}}(A)$).

If $\varepsilon_1 = \varepsilon_2 = \varepsilon$ we refer to $\mathcal{C}$ as an absolute (resp. relative) ε-close approximation of $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$.

Through identity (3.1) we have that an absolute (resp. relative) $(\varepsilon_1, \varepsilon_2)$-close approximation to $\mathsf{FI}\left(\mathcal{D}, \mathcal{I}, f_{\mathcal{D}}^{(K)}\right)$ is an $(\varepsilon_1, \varepsilon_2)$-approximation to $\mathsf{TOPK}(\mathcal{D}, \mathcal{I}, K)$.

For association rules, we generalize Def. 7 as follows.

Definition 11. Given two parameters $\varepsilon_1, \varepsilon_2 \in (0, 1)$ an *absolute* (resp. *relative*) $(\varepsilon_1, \varepsilon_2)$-*close approximation* of $\mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$ is a set

$$\mathcal{C} = \{(W, f_W, \mathcal{K}_W, c_W, \mathcal{J}_W) \mid \text{AR } W, f_W \in \mathcal{K}_W, c_W \in \mathcal{J}_W\}$$

of tuples $(W, f_W, \mathcal{K}_W, c_W, \mathcal{J}_W)$ where f_W and c_W approximate $f_{\mathcal{D}}(W)$ and $c_{\mathcal{D}}(W)$ respectively and belong to $\mathcal{K}_W \subseteq [0, 1]$ and $\mathcal{J}_W \subseteq [0, 1]$ respectively. $\mathcal{C}$ is such that:

1. $\mathcal{C}$ contains all association rules appearing in $\mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$;
2. $\mathcal{C}$ contains no association rule W with frequency $f_{\mathcal{D}}(W) < \theta - \varepsilon_1$ (resp. $f_{\mathcal{D}}(W) < \theta(1 - \varepsilon_1)$);

3. For every tuple $(W, f_W, \mathcal{K}_W, c_W, \mathcal{J}_W) \in \mathcal{C}$, it holds $|f_{\mathcal{D}}(W) - f_W| \leq \varepsilon_2$ and $|\mathcal{K}_W| \leq 2\varepsilon_2$ (resp. $|f_{\mathcal{D}}(W) - f_W| \leq \varepsilon_2 f_D s(W)$ and $|\mathcal{K}_W| \leq 2\varepsilon_2 f_{\mathcal{D}}(W)$).
4. $\mathcal{C}$ contains no association rule W with confidence $c_{\mathcal{D}}(W) < \gamma - \varepsilon_1$ (resp. $c_{\mathcal{D}}(W) < \gamma(1 - \varepsilon_1)$);
5. For every tuple $(W, f_W, \mathcal{K}_W, c_W, \mathcal{J}_W) \in \mathcal{C}$, it holds $|c_{\mathcal{D}}(W) - c_W| \leq \varepsilon_2$ and $|\mathcal{J}_W| \leq 2\varepsilon_2$ (resp. $|c_{\mathcal{D}}(W) - c_W| \leq \varepsilon_2 c_D s(W)$ and $|\mathcal{J}_W| \leq 2\varepsilon_2 c_{\mathcal{D}}(W)$).

If $\varepsilon_1 = \varepsilon_2 = \varepsilon$ we refer to $\mathcal{C}$ as an absolute (resp. relative) ε-close approximation of $\mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$.

It is easy to see that it is possible to modify Lemmas 4, 5, 6, 7, 9, and 8 to return absolute or relative $(\varepsilon, \varepsilon/2)$-close approximations of the relevant collections of FIs or ARs.

4.2.1 MapReduce

MapReduce is a programming paradigm and an associated parallel and distributed implementation for developing and executing parallel algorithms to process massive datasets on clusters of commodity machines [Dean and Ghemawat, 2008]. Algorithms are specified in MapReduce using two functions, **map** and **reduce**. The input is seen as a sequence of ordered key-value pairs (k, v). The **map** function takes as input one such $(key, value)$ pair at a time, and can produce a finite multiset of pairs $\{(k_1, v_1), (k_2, v_2), \cdots\}$. Let $\mathcal{U}$ be the multiset union of all the multisets produced by the **map** function when applied to all input pairs. We can partition $\mathcal{U}$ into sets $\mathcal{U}_{\bar{k}}$ indexed by a particular key $\bar{k}$. $\mathcal{U}_{\bar{k}}$ contains all and only the values v for which there are pairs $(\bar{k}, v)$ with key $\bar{k}$ produced by the function **map** ($\mathcal{U}_{\bar{k}}$ is a multiset, so a particular value v can appear multiple times in $\mathcal{U}_{\bar{k}}$). The **reduce** function takes as input a key $\bar{k}$ and the multiset $\mathcal{U}_{\bar{k}}$ and produce another set $\{(k_1, v_1), (k_2, v_2), \cdots\}$. The output of **reduce** can be used as input for another (different) **map** function to develop MapReduce algorithms that complete in multiple *rounds*. By definition, the **map** function can be executed in parallel for each input pair. In the same way, the computation of the output of **reduce** for a specific key k^* is independent from the computation for any other key $k' \neq k^*$, so multiple copies of the **reduce** function can be executed in parallel, one for each key k. We denote the machines executing the **map** function as *mappers* and those executing the **reduce** function as *reducers*. The latter will be indexed by the key k assigned to them, i.e., reducer r processes the multiset $\mathcal{U}_r$. The data produced by the mappers are split by key and sent to the reducers in the so-called *shuffle* step. Some implementations, including Hadoop and the one described by Google [Dean and Ghemawat, 2008], use sorting in the shuffle step to perform the grouping of map outputs by key. The shuffle is transparent to the algorithm designer but, since it

involves the transmission of (possibly very large amount of) data across the network, can be very expensive.

4.3 Algorithm

In this section we describe and analyze PARMA, our algorithm for extracting ε-close approximations of $\mathsf{FI}(\mathcal{D},\mathcal{I},\theta)$, $\mathsf{TOPK}(\mathcal{D},\mathcal{I},\theta)$, and $\mathsf{AR}(\mathcal{D},\mathcal{I},\theta,\gamma)$ from samples of a dataset $\mathcal{D}$ with probability at least $1-\delta$. In this section we present the variant for the absolute case and $\mathsf{FI}(\mathcal{D},\mathcal{I},\theta)$. The variants for the relative cases and for $\mathsf{TOPK}(\mathcal{D},\mathcal{I},\theta)$ and $\mathsf{AR}(\mathcal{D},\mathcal{I},\theta,\gamma)$ can be easily derived from the one we present here. We outline them in Sect. 4.3.4.

4.3.1 Design

We now present the algorithmic design framework on which we developed PARMA and some design decisions we made for speeding up the computation.

Model When developing solutions for any computational problem, the algorithm designer must always be aware of the trade-off between the available computational resources and the performance (broadly defined) of the algorithm. In the parallel computation setting, the resources are usually modeled through the parameters p and m, representing respectively the number of available processors that can run in parallel and the amount of local memory available to a single processor. In our case we will express m in terms of the number of transactions that can be stored in the main memory of a single machine. When dealing with algorithms that use random samples of the input, the performances of the algorithm are usually measured through the parameters ε and δ. The former represents the desired accuracy of the results, i.e., the maximum tolerable error (defined according to some distance measure) in the solution computed by the algorithm using the random sample when compared to an exact solution of the computational problem. The parameter δ represents the maximum acceptable probability that the previously defined error in the solution computed by the algorithm is greater than ε. The measure we will use to evaluate the performances of PARMA in our analysis is based on the concept of absolute ε-close approximation introduced in Definition 10.

Trade-offs We are presented with a trade-off between the parameters ε, δ, p, and m. To obtain a ε-approximation with probability at least $1-\delta$, one must have a certain amount of computational

resources, expressed by p and m. On the other hand, given p and m, it is possible to obtain absolute ε-close approximations with probability at least $1-\delta$ only for values of ε and δ larger than some limits. By fixing any three of the parameters, it is possible to find the best value for the fourth by solving an optimization problem. From Lemma 4 we know that there is a trade-off between ε, δ, and the size w of a random sample from which it is possible to extract a $(\varepsilon, \varepsilon/2)$-approximation to $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ with probability at least $1-\delta$. If $w \leq m$, then we can store the sample in a single machine and compute the absolute ε-close approximation there using Lemma 4. For some combinations of values for ε and δ, though, we may have that $w > m$, i.e. the sample would be too large to fit into the main memory of a single processor, defeating one of the goals of using random sampling, that is to store the set of transactions to be mined in main memory in order to avoid expensive disk accesses. To address the issue of a single sample not fitting in memory, PARMA works on multiple samples, say N with $N \leq p$, each of size $w \leq m$ so that **1)** each sample fits in the main memory of a single processor and **2)** for each sample, it is possible to extract an absolute $(\varepsilon, \varepsilon/2)$-close approximation of $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ from it with probability at least $1-\phi$, for some $\phi > \delta$. In the first stage, the samples are created and mined in parallel and the so-obtained collections of Frequent Itemset are then aggregated in a second stage to compute the final output. This approach is a perfect match for the MapReduce framework, given the limited number of synchronization and communication steps that are needed. Each stage is performed in a single MapReduce round. The computational and data workflow of PARMA is presented in Figure 4.1, which we describe in detail in the following paragraphs.

Computing N and w From the above discussion it should be clear that, once p, m, ε and δ have been fixed, there is a trade-off between w and N. In the MapReduce setting, often the most expensive operation is the movement of data between the mappers and the reducers in the shuffle phase. In PARMA, the amount of data to be shuffled corresponds to the sum of the sizes of the samples, i.e., Nw, and to the amount of communication needed in the aggregation stage. This second quantity is dependent on the number of frequent itemsets in the dataset, and therefore PARMA has no control over it. PARMA tries to minimize the first quantity when computing N and w in order to achieve the maximum speed. It is still possible to minimize for other quantities (e.g. ε or δ if they have not been fixed), but we believe the most effective and natural in the MapReduce setting is the minimization of the communication. This intuition was verified in our experimental evaluation, where communication proved to be the dominant cost. We can formulate the problem of minimizing

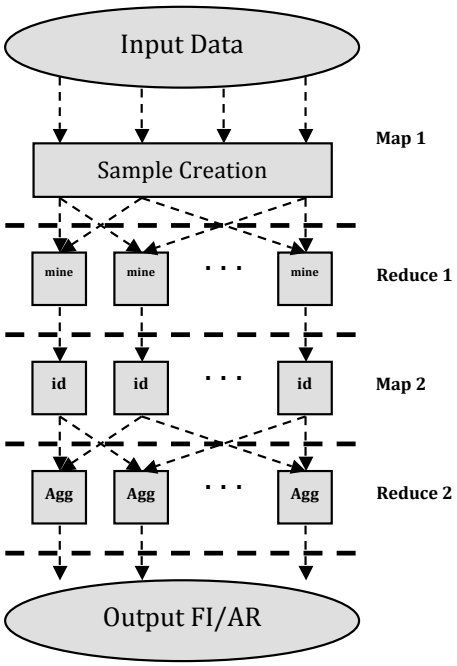


Figure 4.1: A system overview of PARMA. Ellipses represent data, squares represent computations on that data and arrows show the movement of data through the system.

Nw as the following Mixed Integer Non Linear Programming (MINLP) problem:

- **Variables:** non-negative integer N, real $\phi \in (0, 1)$,
- **Objective:** minimize $2N/\varepsilon^2(d + \log(1/\phi))$.
- **Constraints:**

$$N \leq p \tag{4.1}$$

$$\phi \geq e^{-m\varepsilon^2/2+d} \tag{4.2}$$

$$N(1-\phi) - \sqrt{N(1-\phi)2\log(1/\delta)} \geq N/2 + 1 \tag{4.3}$$

Note that, because of our requirement **2)** on w, the sample size w is directly determined by ϕ through Lemma 4, so the trade-off is really between N and ϕ, while w does not appear in the above problem. Since ϕ is a probability we restrict its domain to the interval $(0, 1)$, but it must also be such that the single sample size w is at most m, as required by **1)** and expressed by Constraint (4.2). The limit to the number of samples N is expressed by Constraint (4.1). The last constraint (4.3) is a bit more technical and the need for it will be evident in the analysis of the algorithm. Intuitively, it

expresses the fact that an itemset must appear in a sufficiently high fraction (at least 1/2, possibly more) of the collections obtained from the samples in the first stage in order to be included in the output collection. Due to the integrality constraint on N, this optimization problem is not convex, although when the constraint it is dropped the feasibility region is convex, and the objective function is convex. It is then relatively easy and fast to find an integer optimal solution to the problem using a global MINLP solver like BARON [Sahinidis and Tawarmalani, 2010].

4.3.2 Description

In the following paragraphs we give a detailed description of PARMA. The reader is also referred to Figure 4.1 for a schematic representation of PARMA's data/computational workflow.

Stage 1: Sampling and Local Mining Once ϕ, w and N have been computed, PARMA enters the first MapReduce round to create the N samples (phase Map 1 in Figure 4.1) and mine them (Reduce 1). We see the input of the algorithm as a sequence

$$(1, \tau_1), (2, \tau_2), \cdots, (|\mathcal{D}|, \tau_{|\mathcal{D}|}),$$

where the τ_i are transactions in $\mathcal{D}$. In the Map phase, the input of the **map** function is a pair (tid, τ), where tid is a natural from 1 to $|\mathcal{D}|$ and τ is a transaction in $\mathcal{D}$. The map function produces in output a pair $(i, (\ell_\tau^{(i)}, \tau))$ for each sample $\mathcal{S}_i$ containing τ. The value $\ell_\tau^{(i)}$ denotes the number of times τ appears in $\mathcal{S}_i$, $1 \leq i \leq N$. We use random sampling with replacement and ensure that all samples have size w, i.e., $\sum_{\tau \in \mathcal{D}} \ell_\tau^{(i)} = w$, $\forall i$. This is done by computing (serially) how many transactions each mapper must send to each sample. In the Reduce phase, there are N reducers, with associated key i, $1 \leq i \leq N$. The input to reducer i is $(i, \mathcal{S}_i)$, $1 \leq i \leq N$. Reducer i mines the set $\mathcal{S}_i$ of transactions it receives using an exact sequential mining algorithm like Apriori or FP-Growth and a lowered minimum frequency threshold $\theta' = \theta - \varepsilon/2$ to obtain $\mathcal{C}_i = \mathsf{FI}(\mathcal{S}_i, \mathcal{I}, \theta')$. For each itemset $A \in \mathcal{C}_i$ the Reduce function outputs a pair $(A, (f_{\mathcal{S}_i}(A), [f_{\mathcal{S}_i}(A) - \varepsilon/2, f_{\mathcal{S}_i}(A) + \varepsilon/2])$.

Stage 2: Aggregation In the second round of MapReduce, PARMA aggregates the result from the first stage to obtain an absolute ε-close approximation to $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ with probability at least $1 - \delta$. The Map phase (Map 2 in Figure 4.1) is just the identity function, so for each pair

$$(A, (f_{\mathcal{S}_i}(A), [f_{\mathcal{S}_i}(A) - \varepsilon/2, f_{\mathcal{S}_i}(A) + \varepsilon/2])$$

in the input the same pair is produced in the output. In the Reduce phase (Reduce 2) there is a reducer for each itemset A that appears in at least one of the collections $\mathcal{C}_j$ (i.e., $\forall A$ such that there is a $\mathcal{C}_j$ containing a pair related to A). The reducer receives as input the itemset A and the set $\mathcal{F}_A$ of pairs

$$(f_{\mathcal{S}_i}(A), [f_{\mathcal{S}_i}(A) - \varepsilon/2, f_{\mathcal{S}_i}(A) + \varepsilon/2])$$

for the samples $\mathcal{S}_i$ such that $A \in \mathcal{C}_i$. Now let

$$R = N(1-\phi) - \sqrt{N(1-\phi)2\log(1/\delta)}. \tag{4.4}$$

The itemset A is declared *globally frequent* and will be present in the output if and only if $|\mathcal{F}_A| \geq R$. If this is the case, PARMA computes, during the Reduce phase of the second MapReduce round, the estimation $\tilde{f}(A)$ for the frequency $f_{\mathcal{D}}(A)$ of the itemset A in $\mathcal{D}$ and the confidence interval $\mathcal{K}_A$. The computation for $\tilde{f}(A)$ proceeds as follows. Let $[a_A, b_A]$ be the *shortest* interval such that there are at least $N - R + 1$ elements from $\mathcal{F}_A$ that belong to this interval. The estimation $\tilde{f}(A)$ for the frequency $f_{\mathcal{D}}(A)$ of the itemset A is the central point of this interval:

$$\tilde{f}(A) = a_A + \frac{b_A - a_A}{2}$$

The confidence interval $\mathcal{K}_A$ is defined as

$$\mathcal{K}_A = \left[a_A - \frac{\varepsilon}{2}, b_A + \frac{\varepsilon}{2}\right].$$

The output of the reducer assigned to the itemset A is

$$(A, (\tilde{f}(A), \mathcal{K}_A)).$$

The output of PARMA is the union of the outputs from all reducers.

4.3.3 Analysis

We have the following result:

Lemma 12. *The output of the PARMA is an absolute ε-close approximation of $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ with probability at least $1 - \delta$.*

Proof. For each sample $\mathcal{S}_i$, $1 \leq i \leq N$ we define a random variable X_i that takes the value 1 if $\mathcal{C}_i = \mathsf{FI}(\mathcal{S}_i, \mathcal{I}, \theta')$ is an absolute $(\varepsilon, \varepsilon/2)$-close approximation of $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$, $X_i = 0$ otherwise. Given

our choices of w and θ', we can apply Lemma 4 and have that $\Pr(X_i = 1) \geq 1-\phi$. Let $Y = \sum_{r=1}^{N} X_r$ and let Z be a random variable with binomial distribution with parameters N and $1-\phi$. For any constant $Q < N(1-\phi)$ we have

$$\Pr(Y \leq Q) \leq \Pr(Z \leq Q) \leq e^{-N(1-\phi)(1-\frac{Q}{N(1-\phi)})^2/2},$$

where the last inequality follows from an application of the Chernoff bound [Mitzenmacher and Upfal, 2005, Chap. 4]. We then have, for our choice of ϕ and N and for $Q = R$ (defined in Eq. (4.4)), that with probability at least $1-\delta$, at least R of the collections $\mathcal{C}_i$ are absolute $(\varepsilon, \varepsilon/2)$-close approximations of $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$. Denote this event as $\mathcal{G}$. For the rest of the proof we will assume that $\mathcal{G}$ indeed occurs.

Then $\forall A \in \mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$, A belongs to at least R of the collections $\mathcal{C}_i$, therefore a triplet $(A, \tilde{f}(A), \mathcal{K}_A)$ will be in the output of the algorithm. This means that Property 1 from Def. 10 holds.

Consider now any itemset B such that $f_{\mathcal{D}}(B) < \theta - \varepsilon$. By definition of absolute $(\varepsilon, \varepsilon/2)$-close approximation we have that B can only appear in the collections $\mathcal{C}_i$ that are not absolute $(\varepsilon, \varepsilon/2)$-close approximations. Given that $\mathcal{G}$ occurs, then there are at most $N - R$ such collections. But from Constraint (4.3) and the definition of R in (4.4), we have that $N - R < R$, and therefore B will not be present in the output of PARMA, i.e. Property 2 from Def. 10 holds.

Let now C be any itemset in the output, and consider the interval $S_C = [a_C, b_C]$ as computed by PARMA. S_C contains at least $N - R + 1$ of the $f_{\mathcal{S}_i}(C)$, otherwise C would not be in the output. By our assumption on the event $\mathcal{G}$, we have that at least R of the $f_{\mathcal{S}_i}(C)$'s are such that $|f_{\mathcal{S}_i}(C) - f_{\mathcal{D}}(C)| \leq \varepsilon/2$. Then there is an index j such that $|f_{\mathcal{S}_j}(C) - f_{\mathcal{D}}(C)| \leq \varepsilon/2$ and such that $f_{\mathcal{S}_j}(C) \in S_C$. Given also that $f_{\mathcal{S}_j}(C) \geq a_C$, then $f_{\mathcal{D}}(C) \geq a_C - \varepsilon/2$, and analogously, given that $f_{\mathcal{S}_j}(C) \leq b_C$, then $f_{\mathcal{D}}(C) \leq b_C + \varepsilon/2$. This means that

$$f_{\mathcal{D}}(C) \in \left[a_C - \frac{\varepsilon}{2}, b_C + \frac{\varepsilon}{2}\right] = \mathcal{K}_C, \tag{4.5}$$

which, together with the fact that $\tilde{f}_C \in \mathcal{K}_C$ by construction, proves Property 3.b from Def. 10. We now give a bound to $|S_C| = b_C - a_C$. From our assumption on the event $\mathcal{G}$, there are (at least) R values $f_{\mathcal{S}_i}(C)$ such that $|f_{\mathcal{S}_i}(C) - f_{\mathcal{D}}(C)| \leq \varepsilon/2$, then the interval $[f_{\mathcal{D}}(C) - \varepsilon/2, f_{\mathcal{D}}(C) + \varepsilon/2]$ contains (at least) R values $f_{\mathcal{S}_i}(C)$. Its length ε is an upper bound to $|S_C|$. Then the length of the interval $\mathcal{K}_C = [a_C - \varepsilon/2, b_C + \varepsilon/2]$ is at most 2ε, as requested by Property 3.c from Def. 10. From this, from (4.5), and from the fact that $\tilde{f}(C)$ is the center of this interval we have $|\tilde{f}(C) - f_{\mathcal{D}}(C)| \leq \varepsilon$, i.e., Property 3.a from Def. 10 holds. □

4.3.4 Top-K Frequent Itemsets and Association Rules

The above algorithm can be easily adapted to computing, with probability at least $1-\delta$, absolute and relative ε-close approximations to $\mathsf{TOPK}(\mathcal{D}, \mathcal{I}, K)$ and to $\mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$. The main difference is in the formula to compute the sample size w (Lemma 4), and in the process to extract the local

collections from the samples. The case of top-K is presented in Lemma 6, while for the association rule case we can use Lemma 9. These are minimal changes to the version of PARMA presented here, and the modified algorithms guarantee the same levels of accuracy and confidence.

4.4 Implementation

The entire PARMA algorithm has been written as a Java library for Hadoop, the popular open source implementation of MapReduce. Because all experiments were done using Amazon Web Service (AWS) Elastic MapReduce, the version of Hadoop used was 0.20.205, the highest supported by AWS. The use of Java makes possible future integration with the Apache Mahout parallel machine learning library (`https://mahout.apache.org`). Mahout also includes an implementation of PFP [Li et al., 2008] that we used for our evaluation of PARMA.

In PARMA, during the mining phase (i.e. during the reducer of stage 1), any frequent itemset or association rule mining algorithm can be used. We wanted to compare the performances of PARMA against PFP which only produces frequent itemsets, therefore we chose to use a frequent itemset mining algorithm instead of an association rule mining algorithm. Again, this choice was merely for ease of comparison with existing parallel frequent itemset mining algorithms as no such algorithms for association rule mining exist. While there are many frequent itemset mining algorithms available, we chose the FP-growth implementation provided by [Coenen, 2014]. We chose FP-growth due to its relative performance superiority to other Frequent Itemsets mining algorithms. Additionally, since FP-growth is the algorithm that PFP has parallelized and uses internally, the choice of FP-growth for the mining phase in PARMA is appropriate for a more natural comparison.

We also compare PARMA against the naive distributed counting algorithm (DistCount) for computing frequent itemsets. In this approach, there is only a single MapReduce iteration. The map breaks a transaction τ into its powerset $\mathcal{P}(\tau)$ and emits key/value pairs in the form $(A, 1)$ where A is an itemset in $\mathcal{P}(t)$. The reducers simply count how many pairs they receive for each itemset A and output the itemsets with frequency above the minimum frequency threshold. This is similar to the canonical wordcount example for MapReduce. However, because the size of the powerset is exponential in the size of the original transaction (specifically $2^{|\tau|}$, where $|\tau|$ denotes the number of items in a given transaction), this algorithm incurs massive network costs, even when combiners are used. This is very similar to the algorithms presented in [Cryans et al., 2010; Li and Zhang, 2011;

Yang et al., 2010]. We have built our own implementation of DistCount in Java using the Hadoop API.

4.5 Experimental evaluation

We evaluate the performance of PARMA using Amazon's Elastic MapReduce platform. We used instances of type *m1.xlarge*, which contain roughly 17GB of memory and 6.5 EC2 compute units. For data, we created artificial dataset using the synthetic data generator from [Cristofor, 2006]. This implementation is based on the generator described in [Agrawal and Srikant, 1994], which can be parameterized to generate a wide range of data. We used two distinct sets of parameters to generate the datasets: the first set, shown in Table 4.1, for the experiments comparing PARMA and the distributed counting algorithm (DistCount), and the second set, shown in Table 4.2, for the experiments comparing PARMA and PFP. The parameters were chosen to mimic real-world datasets on which PARMA would be run. For a full description of the relevant parameters, we refer the reader to [Agrawal and Srikant, 1994]. The reason we needed two distinct datasets is that DistCount did not scale to the larger dataset sizes, as the amount of data it generates in the map phase grows exponentially with the length of the individual transactions in the dataset. We found that DistCount would run out of memory using datasets with longer transactions, and we had to generate datasets with both shorter and less transactions for its comparisons.

Because PARMA is an approximate algorithm, the choice of accuracy parameters ε and δ are important, as is θ, the minimum frequency at which itemsets were mined. In all of our experiments, $\varepsilon = 0.05$ and $\delta = 0.01$. This means that the collection of itemsets mined by PARMA will be an absolute 0.05-close approximation with probability 0.99. In practice, we show later that the results are much more accurate than what this. For all experiments other than the minimum frequency performance comparison in Figure 4.4 and for the accuracy comparison in Figures 4.6 and 4.7, θ was kept constant at 0.1.

We focus on three main aspects in the experimental analysis of PARMA. First is the runtime analysis. In this set of experiments we compare PARMA to PFP and DistCount using several different datasets of varying size and mined with varying minimum frequency thresholds. We then break down the runtime of PARMA into the map, reduce and shuffle phases of each of the two stages. This demonstrates which parts of the algorithm have runtimes that increase as the data size

number of items	1000
average transaction length	5
average size of maximal potentially large itemsets	5
number of maximal potentially large itemsets	5
correlation among maximal potentially large itemsets	0.1
corruption of maximal potentially large itemsets	0.1

Table 4.1: Parameters used to generate the datasets for the runtime comparison between DistCount and PARMA in Figure 4.2.

number of items	10000
average transaction length	10
average size of maximal potentially large itemsets	5
number of maximal potentially large itemsets	20
correlation among maximal potentially large itemsets	0.1
corruption of maximal potentially large itemsets	0.1

Table 4.2: Parameters used to generate the datasets for the runtime comparison between PFP and PARMA in Figure 4.2.

grows and which parts are relatively data independent. The second aspect is speedup, to show the scalability of our algorithm. In these experiments, data and cluster size are varied to determine how PARMA scales. Finally, because the output of PARMA is an absolute ε-close approximation of the real set of frequent itemsets, we provide an accuracy analysis to verify that our results are indeed within the desired quality bounds.

4.5.1 Performance analysis

For the performance analysis of PARMA, we analyze the relative performance against two exact FIM algorithms on MapReduce, *DistCount* and *PFP*, on a cluster of 8 nodes. We also provide a breakdown of the costs associated with each stage of PARMA.

Figure 4.2 (top) shows the comparison between PARMA and DistCount. As discussed previously, due to limitations in the scalability of DistCount, we were unable to test on the larger datasets, so the smaller datasets were generated using parameters from Table 4.1. For DistCount, longer itemsets affect runtime the most, as the number of key/value pairs generated from each transaction is exponential in the size of the transaction. This is not to say that more transactions does not affect runtime, just that the length of those transactions also has a significant impact. Because of this, it is possible to have datasets with fewer transactions but with more "long" transactions that take longer to mine. This effect is seen in the first three datasets (1-3 million). Even though the number of transactions significantly increases, the relative length of the longest transactions was actually

longer in the 1 million transaction dataset. Indeed, upon further inspection, we found that the 1 million transaction dataset had 25 transactions over 20 items in length, while the 2 and 3 million transaction dataset had less than 15. Of course, since these datasets were generated independently and with the same parameters, this was purely by chance. However, as the number of transactions continues to increase, the exponential growth in the number of intermediate key/value pairs is seen by the sharp increase in runtime. While we tried to test with a dataset with 6 million transaction, DistCount ran out of memory. The lack of ability to handle either long individual transactions or a large number of transactions in a dataset limits DistCount's real-world applicability. The runtime of PARMA is significantly faster than that of DistCount and, more importantly, nearly constant across dataset sizes. The reasons for this will be discussed in detail later.

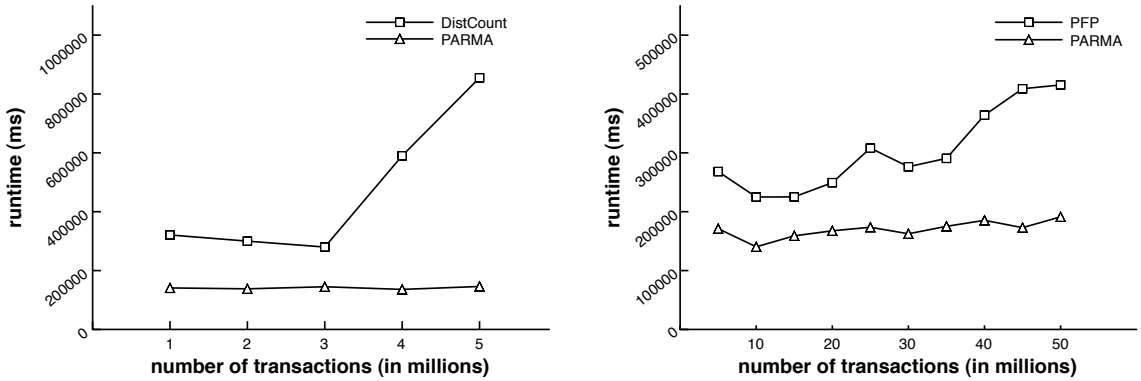


Figure 4.2: A runtime comparison of PARMA with DistCount and PFP.

For the performance comparison with PFP, 10 datasets were generated using parameter values from Table 4.2 and ranging in size from 10 to 50 million transactions. The results are shown in Figure 4.2 (bottom). For every dataset tested, PARMA was able to mine the dataset roughly 30-55% faster than PFP. The reason for the relative performance advantage of PARMA is twofold. The first (and primary) reason is that for larger datasets the size of the dataset that PARMA has sampled (and mined) is staying the same, whereas PFP is mining more and more transactions as the dataset grows. The second reason is that as the dataset grows, PFP is potentially duplicating more and more transactions as it assigns transactions to groups. A transaction that belongs to multiple groups is sent to multiple reducers, resulting in higher network costs.

The most important aspect of the comparison of PFP to PARMA is that the runtimes as data grows are clearly diverging due to the reasons discussed above. While 50 million transactions is very

sizable, it is not hard to imagine real-world datasets with transactions on the order of billions. Indeed, many point-of-sale datasets would easily break this boundary. In these scenarios a randomized algorithm such as PARMA would show increasing performance advantages over exact algorithms such as any of the standard non-parallel algorithms or PFP, which must mine the entire dataset. At that scale, even transmitting that data over the network (several times in the case of PFP) would become prohibitive.

To understand the performance of PARMA it is important to analyze the runtimes at each of the various stages in the algorithm. To do this, we have implemented runtime timers at very fine granularities throughout our algorithm. The timers' values are written to Hadoop job logs for analysis. This breakdown allows us to not only analyze the overall runtime, but also the sections of the algorithm whose runtimes are affected by an increase in data size. In Figure 4.3, a breakdown of PARMA runtimes is shown for each of the six segments of the algorithm, which include a map, shuffle and reduce phase for each of the two stages. Due to space limitations, we only show the breakdown for a subset of the datasets we tested. We observed the same patterns for all datasets. This breakdown demonstrates several interesting aspects of PARMA. First, the cost of the mining local frequent itemsets (stage 1, reduce) is relatively constant. For many frequent itemset mining implementations, this cost will grow with the size of the input. This is not the case in PARMA, because local frequent itemset mining is being done on constant-sized sample of the input. Indeed another interesting observation, as expected, is that the only cost that increases as sample size increases is the cost of sampling (stage 1, map). This is because in order to be sampled the input data must be read, so larger input data means larger read times. In practice, this cost is minimal and grows linearly with the input, hence it will never be prohibitive, especially considering all other current algorithms must read the entire input data at least once, and in many cases multiple times.

There is one outlier in the graph, which is the dataset with 5 million transactions. Because each dataset was independently generated, it is possible for a dataset to have a larger number of frequent itemsets than other datasets, even if it has less transactions. This is the case with the 5 million transaction dataset, which takes longer to run mine for both PARMA and PFP due to the relatively greater number of frequent itemsets.

Figure 4.4 shows the breakdown of PARMA runtimes as the minimum frequency at which the data is mined at is changed. Data size was kept constant at 10 million transactions. Minimum frequency is used by the local frequent itemset mining algorithm to prune itemsets; itemsets below

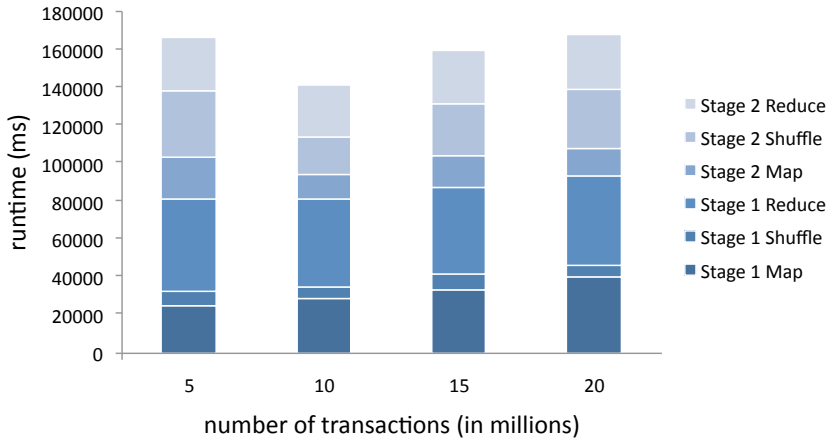


Figure 4.3: A comparison of runtimes of the map/reduce/shuffle phases of PARMA, as a function of number of transactions. Run on an 8 node Elastic MapReduce cluster.

the minimum frequency are not considered frequent, nor is any superset since a superset must, by definition, contain the not frequent set and therefore cannot be frequent itself. Intuitively, a lower minimum frequency will mean more frequent itemsets are produced. Other than a runtime increase in the local frequent itemset mining phase (stage 1, reduce), the effects of this can be seen in the stage 2 shuffle phase as well, as there is more data to move across the network. Still, the added costs of mining with lower frequencies are relatively small.

4.5.2 Speedup and scalability

To show the speedup of PARMA, we used a two-nodes cluster as the baseline. Because PARMA is intended to be a parallel algorithm, the choice of a two-nodes cluster was more appropriate than the standard single node baseline. For the dataset, we used a 10 million transaction database generated using the parameters in Table 4.2. The results are shown in Figure 4.5a. The three lines on this graph represent the relative speedup of both stage 1 and stage 2 as well as the overall PARMA algorithm. The graph indicates that stage 1 is highly parallelizable and follows a near-ideal speedup for up to 8 nodes, after which a slight degradation of speedup occurs. There are two reasons for this slight degradation. In the map phase of stage 1, due to an implementation decision in Hadoop, the smallest unit of data that can be split is one HDFS block. As we continue to add more nodes to the cluster, we may have more available map slots than HDFS data blocks, resulting in some slots being unused. Theoretically, this could be fixed by allowing smaller granularity splitting in

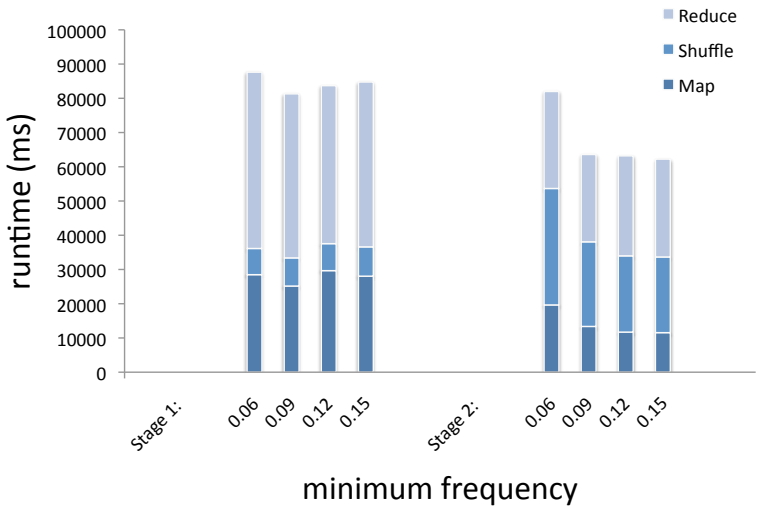
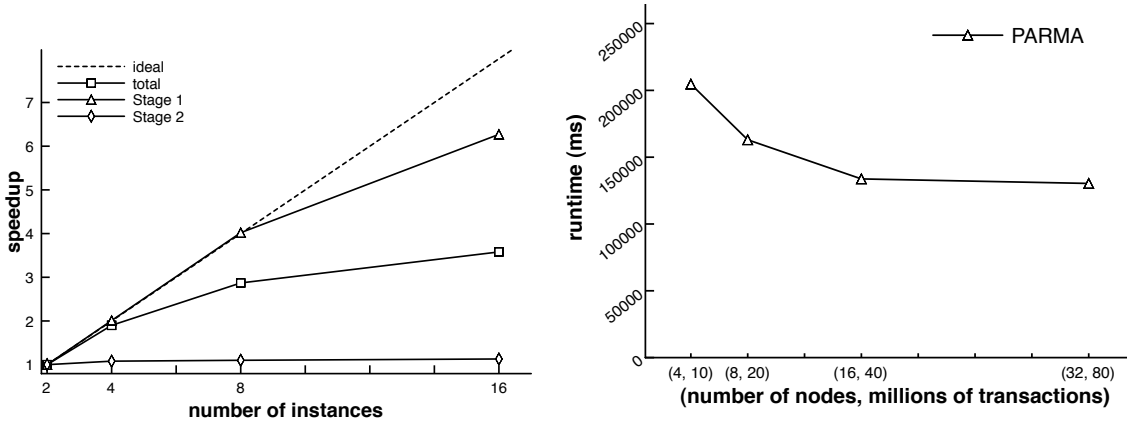


Figure 4.4: A comparison of runtimes of the map/reduce/shuffle phases of PARMA, as a function of minimum frequency. Clustered by stage. Run on an 8 node Elastic MapReduce cluster.

Hadoop. Another cause of the slightly sub-ideal speedup in stage 1 is from the reducer. Because the data in this experiment was held constant, the slight degradation in speedup as more than 8 nodes were added was a result of an inefficient over-splitting of transaction data. If each reducer in stage 1 is mining a very small subset of the transactions, the overhead of building the FP-tree begins to dominate the cost of mining the FP-tree. This is because the cost of mining the FP-tree is relatively fixed. Thus, we can "over-split" the data by forcing the reducer to build a large FP-tree only to mine a small set of transactions. For larger samples, the size of the cluster where speedup degradation begins to occur would also increase, meaning PARMA would continue to scale.

Also, as is clearly visible in the graph, the sub-ideal overall speedup is due largely to the poor speedup of stage 2. Stage 2 is bound almost entirely by the communication costs of transmitting the local frequent itemsets from stage 1 to the reducers that will do the aggregation. Because the amount of local frequent itemsets does not change as more nodes are added, the communication for this stage does not change. What does change is the number of itemsets each node must aggregate. During the reduce phase, each node is assigned a set of keys. All key/value pairs emitted from the map phase are sent to the reducer assigned their respective key. The reducer is in charge of aggregating the values and emitting one aggregate value per key assigned to it. As more reducers are added to the cluster, each reducer will have fewer keys assigned to it, and therefore must aggregate



(a) Speedup analysis broken down by stages.

(b) Scalability as both data and cluster size are increased.

Figure 4.5: Speedup and scalability.

across fewer values, resulting in faster aggregation. The small but existent positive change in the line for stage 2 is a result of this slight speedup of the reduce phase.

Figure 4.5b depicts the scalability of PARMA as both the size of the dataset (i.e. number of transactions) and the number of nodes in the cluster are increased. The data and nodes are scaled proportionally so that the ratio of data to nodes remains constant across all experiments. This result shows that as nodes and data are increased proportionally, the total runtime actually begins to decrease for larger datasets. This is because as nodes are added to the cluster, the runtime of the Stage 1 reducer (FIM) is decreased while the relative costs of the Stage 1 mapper and Stage 2 remain the same. There is a leveling off of the runtime between the 40M and 80M datasets, which can be explained using Amdhal's law; because only portions of the algorithm are parallelizable, there is a theoretical maximum speedup that is possible. Still, the constant runtime as data is increased demonstrates PARMA's potential scalability to real-world cluster and dataset sizes.

A very important aspect of PARMA that should be stressed again here is that the size of the sample that will be mined does not depend directly on size of the original database, but instead on the confidence parameters ε and δ. From a practical perspective, this means that assuming confidence parameters are unchanged, larger and larger datasets can be mined with very little increase in overall runtime (the added cost will only be the extra time spent reading the larger dataset initially during sampling). Because of this, clusters do not need to scale with the data, and often a relatively modest cluster size will be able to mine itemsets in a very reasonable time.

4.5.3 Accuracy

The output of PARMA is a collection of frequent itemsets which approximates the collection one can obtain by mining the entire dataset. Although our analysis shows that PARMA offers solid guarantees in terms of accuracy of the output, we conducted an extensive evaluation to assess the actual performances of PARMA in practice, especially in relation to what can be analytically proved.

We compared the results obtained by PARMA with the exact collection of itemsets from the entire dataset, for different values of the parameters ε, δ, and θ, and for different datasets. A first important result is that in all the runs, the collection computed by PARMA was indeed an absolute ε-close approximation to the real one, i.e., all the properties from Definition 10 were satisfied. This fact suggests that the confidence in the result obtained by PARMA is actually greater than the level $1-\delta$ suggested by the analysis. This can be explained by considering that we had to use potentially loose theoretical bounds in the analysis to make it tractable.

Given that all real frequent itemsets were included in the output, we then focused on how many itemsets with real frequency in the interval $[\theta-\varepsilon, \theta)$ were included in the output. It is important to notice that these itemsets would be *acceptable* false positives, as Definition 10 does not forbid them to be present in the output. We stress again that the output of PARMA never contained *non-acceptable* false positives, i.e. itemsets with real frequency less than the minimum frequency threshold θ. The number of acceptable false positives included in the output of PARMA depends on the distribution of the real frequencies in the interval $[\theta-\varepsilon, \theta)$, so it should not be judged in absolute terms. In Table 4.3 we report, for various values of θ, the number of real frequent itemsets (i.e., with real frequency at least θ, the number of acceptable false positives (AFP) contained in the output of PARMA, and the number of itemsets with real frequency in the interval $[\theta-\varepsilon, \theta)$, i.e., the maximum number of acceptable false positives that may be contained in the output of PARMA (Max AFP). These numbers refers to a run of PARMA on (samples of) the 10M dataset, with $\varepsilon = 0.05$ and $\delta = 0.01$. It is evident that PARMA does a very good job in filtering out even acceptable false positives, especially at lower frequencies, when their number increases. This is thanks to the fact that an itemset is included in the output of PARMA if and only if it appears in the majority of the collections obtained in the first stage. Itemsets with real frequencies in $[\theta-\varepsilon, \theta)$ are not very likely to be contained in many of these collections.

We conducted an evaluation of the accuracy of two other components of the output of PARMA,

θ	Real FI's	Output AFP's	Max AFP's
0.06	11016	11797	201636
0.09	2116	4216	10723
0.12	1367	335	1452
0.15	1053	299	415

Table 4.3: Acceptable False Positives in the output of PARMA

namely the estimated frequencies for the itemsets in the output and the width of the confidence bounds for these estimations. In Figure 4.6 we show the distribution of the absolute error in the estimation, i.e. $|\tilde{f}(X) - f_{\mathcal{D}}(X)|$ for all itemsets X in the output of PARMA, as θ varies. The lower end of the "whisker" indicates the minimum error, the lower bound of the box corresponds to the first quartile, the segment across the box to the median, and the upper bound of the box to the third quartile. The top end of the whisker indicates the maximum error, and the central diamond shows the mean. This figure (and also Figure 4.7) shows the values for a run of PARMA on samples of the 10M dataset, with $\varepsilon = 0.05$ and $\delta = 0.01$. We can see that even the maximum values are one order of magnitude smaller than the threshold of 0.05 guaranteed by the analysis, and many of the errors are two or more orders of magnitude smaller. It is also possible to appreciate that the distribution of the error would be heavily concentrated in a small interval if the maximum error were not so high, effectively an outlier. The fact that the average and the median of the error, together with the entire "box" move down as the minimum frequency threshold decrease can be explained by the fact that at lower frequencies more itemsets are considered, and this makes the distribution less susceptible to outliers. Not only this is a sign of the high level of accuracy achieved by PARMA, but also of its being consistently accurate on a very large portion of the output.

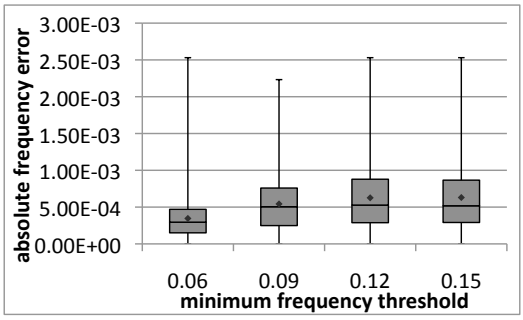


Figure 4.6: Error in frequency estimations as frequency varies.

Finally, in Figure 4.7 we show the distribution of the widths of the confidence intervals $\mathcal{K}(A)$ for the frequency estimations $\tilde{f}(A)$ of the itemsets A in the output of PARMA. Given that $\varepsilon = 0.05$,

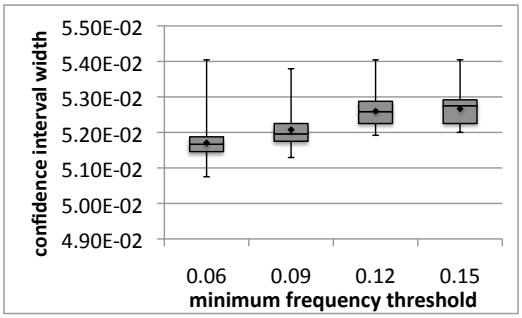


Figure 4.7: Width of the confidence intervals as frequency varies.

the maximum allowed width was $2\varepsilon = 0.1$. It is evident from the figures that PARMA returns much narrower intervals, of size almost ε. Moreover, the distribution of the width is very concentrated, as shown by the small height of the boxes, suggesting that PARMA is extremely consistent in giving very high quality confidence intervals for the estimations. We state again that in all runs of PARMA in our tests, all the confidence intervals contained the estimation and the real frequency, as requested by Definition 10. As seen in the case of the estimation error, the distribution of the widths shifts down at lower thresholds θ. This is motivated by the higher number of itemsets in the output of PARMA at those frequencies. Their presence makes the distribution more robust to outliers. We can conclude that PARMA gives very narrow but extremely accurate confidence intervals across the entirety of its output.

This analysis of the accuracy of various aspects of PARMA's output shows that PARMA can be very useful in practice, and the confidence of the end user in the collections of itemsets and estimations given in its output can be even higher than what is guaranteed by the analysis.

4.6 Conclusions

In this chapter we described PARMA, a parallel algorithm for mining quasi-optimal collections of frequent itemsets and association rules in MapReduce. We showed through theoretical analysis that PARMA offers provable guarantees on the quality of the output collections. Through experimentation on a wide range of datasets ranging in size from 5 million to 50 million transactions, we have demonstrated a 30-55% runtime improvement over PFP, the current state-of-the-art in exact parallel mining algorithms on MapReduce. Empirically we were able to verify the accuracy of the theoretical bounds, as well as show that in practice our results are orders of magnitude more accurate than is

analytically guaranteed. Thus PARMA is an algorithm that can scale to arbitrary data sizes while simultaneously providing nearly perfect results. Indeed PARMA proved its practical usefulness as it was adapted to a streaming setting and integrated by Yahoo in its SAMOA platform for massive distributed streaming computation (`http://yahoo.github.io/samoa/`).

Chapter 5

Finding the True Frequent Itemsets

In this chapter we present the task of mining Frequent Itemsets (FIs) from a different point of view, and show that we can use VC-dimension to avoid the inclusion of *false positives* in the results of mining. In most applications, the set of FIs is not interesting *per se*. Instead, the mining results are used to infer properties of the *underlying process* that generated the dataset. Consider for example the following scenario: a researcher would like to identify frequent associations (i.e., itemsets) between preferences among Facebook users. To this end, she sets up an online survey which is filled out by a *small fraction* of Facebook users (some users may even take the survey multiple times). Using this information, the researcher wants to infer the associations (itemsets) that are frequent for the *entire* Facebook population. In fact, the whole Facebook population and the online survey define the underlying *process* that generated the dataset *observed* by the researcher. We are interested in answering the following question: how can we use the latter (the observed dataset) to identify itemsets that are frequent in the former (the whole population)? This is a very natural question, as is the underlying assumption that the observed dataset is *representative* of the generating process. For example, in market basket analysis, the observed purchases of customers are used to infer the future purchase habits of all customers while assuming that the purchase behavior that generated the dataset is representative of the one that will be followed in the future.

A natural and general model to describe these concepts is to assume that the transactions in

This chapter is an extended version of a work originally appeared in the proceedings of SDM'14 as [Riondato and Vandin, 2014].

the dataset $\mathcal{D}$ are *independent identically distributed* (i.i.d.) samples from an *unknown* probability distribution π defined on all possible transactions built on a set of items. Since π is fixed, each itemset A has a fixed *probability* $t_\pi(A)$ to appear in a transaction sampled from π. We call $t_\pi(A)$ the *true frequency* of A (w.r.t. π). The true frequency corresponds to the fraction of transactions that contain the itemset A among an infinite set of transactions. The real goal of the mining process is then to identify itemsets that have true frequency t_π at least θ, i.e., the *True Frequent Itemsets* (TFIs). In the market basket analysis example, $\mathcal{D}$ contains the observed purchases of customers, the *unknown* distribution π describes the purchase behavior of the customers as a whole, and we want to analyze $\mathcal{D}$ to find the itemsets that have probability (i.e., true frequency) at least θ to be bought by a customer.

Since $\mathcal{D}$ represents only a *finite* sample from π, the set F of frequent itemsets of $\mathcal{D}$ w.r.t. θ only provides an *approximation* of the True Frequent Itemsets: due to the stochastic nature of the generative process, the set F may contain a number of *false positives*, i.e., itemsets that appear among the frequent itemsets of $\mathcal{D}$ but whose *true* frequency is smaller than θ. At the same time, some itemsets with true frequency greater than θ may have a frequency in $\mathcal{D}$ that is *smaller* than θ (*false negatives*), and therefore not be in F. This implies that one can not aim at identifying *all and only* the itemsets having true frequency at least θ. Even worse, from the data analyst's point of view, there is *no guarantee or bound on the number of false positives* reported in F. Consider the following scenario as an example. Let A and B be two (disjoint) sets of pairs of items. The set A contains 1,000 disjoint pairs, while B contains 10,000 disjoint pairs. Let π be such that, for any pair $(a, a') \in A$, we have $t_\pi((a, a')) = 0.1$, and for any pair $(b, b') \in B$, we have $t_\pi((b, b')) = 0.09$. Let $\mathcal{D}$ be a dataset of 10,000 transactions sampled from π. We are interested in finding pairs of items that have true frequency at least $\theta = 0.095$. If we extract the pairs of items with frequency at least θ in $\mathcal{D}$, it is easy to see that in expectation 50 of the 1,000 pairs from A will have frequency in $\mathcal{D}$ *below* 0.095, and in expectation 400 pairs from B will have frequency in $\mathcal{D}$ *above* 0.095. Therefore, the set of pairs that have frequency at least θ in $\mathcal{D}$ does *not* contain some of the pairs that have true frequency at least θ (false negatives), but includes a huge number of pairs that have true frequency smaller than θ (false positives).

In general, one would like to avoid false positives and at the same time find as many TFIs as possible. These are somewhat contrasting goals, and care must be taken to achieve a good balance between them. A naïve but *overly conservative* method to avoid false positives involves the use of

Chernoff and union bounds [Mitzenmacher and Upfal, 2005]. Given an itemset A in $\mathcal{D}$, the quantity $|\mathcal{D}|f_{\mathcal{D}}(A)$ is a random variable with Binomial distribution $\mathcal{B}(|\mathcal{D}|, t_\pi(A))$. It is possible to use standard methods like the Chernoff and the union bounds to bound the deviation of the frequencies in the dataset of *all* itemsets from their expectations. These tools can be used to compute a value $\hat{\theta}$ such that the probability that a non-true frequent itemset B has frequency greater or equal to $\hat{\theta}$ is at most $1-\delta$, for some $\delta \in (0,1)$. This method has the following serious drawback: in order to achieve such guarantee, it is *necessary* to bound the deviation of the frequencies of *all itemsets possibly appearing in the dataset* [Kirsch et al., 2012]. This means that, if the transactions are built on a set of n items, the union bound must be taken over all $2^n - 1$ potential itemsets, even if some or most of them may appear with very low frequency or not at all in samples from π. As a consequence, the chosen value of $\hat{\theta}$ is extremely *conservative*, despite being sufficient to avoid the inclusion of false positives in mining results. The collection of itemsets with frequency at least $\hat{\theta}$ in $\mathcal{D}$, although consisting (probabilistically) only of TFIs, it only contains a *very small* portion of them, due to the overly conservative choice of $\hat{\theta}$. (The results of our experimental evaluation in Sect. 5.5 clearly show the limitations of this method.) More refined algorithms are therefore needed to achieve the correct balance between the contrasting goals of avoiding false positives and finding as many TFIs as possible.

Our contributions. The contributions we make are the following:

- We formally define the problem of mining the *True Frequent Itemsets* w.r.t. a minimum threshold θ, and we develop and analyze an algorithm to *identify a value $\hat{\theta}$ such that, with probability at least $1-\delta$, all itemsets with frequency at least $\hat{\theta}$ in the dataset have true frequency at least θ*. Our method is completely *distribution-free*, i.e., it does not make *any* assumption about the unknown generative distribution π. By contrast, existing methods to assess the significance of frequent patterns after their extraction require a well specified, limited generative model to characterize the significance of a pattern. When additional information about the distribution π is available, it can be incorporated in our method to obtain even higher accuracy.

- We analyze our algorithm using results from *statistical learning theory* and *optimization*. We define a range space associated to a collection of itemsets and give an upper bound to its (empirical) VC-dimension and a procedure to compute this bound, showing an interesting connection with the Set-Union Knapsack Problem (SUKP) [Goldschmidt et al., 1994]. To the

best of our knowledge, we are the first to apply these techniques to the field of TFIs, and in general the first application of the sample complexity bound based on *empirical* VC-dimension to the field of data mining.

- We implemented our algorithm and assessed its performances on simulated datasets with properties – number of items, itemsets frequency distribution, etc.– similar to real datasets. We computed the fraction of TFIs contained in the set of frequent itemsets in $\mathcal{D}$ w.r.t. $\hat{\theta}$, and the number of false positives, if any. The results show that the algorithm is even *more accurate* than the theory guarantees, since *no false positive* is reported in any of the many experiments we performed, and moreover allows the *extraction of almost all TFIs*. We also compared the set of itemsets computed by our method to those obtained with the "Chernoff and union bounds" method presented in the introduction, and found that our algorithm *vastly outperforms* it.

the firstwork:

Outline. In Sect. 5.1 we review relevant previous contributions. Sections 5.2 and 5.3 contain preliminaries to formally define the problem and key concepts that we will use throughout the work. Our proposed algorithm is described and analyzed in Sect. 5.4. We present the methodology and results of our experimental evaluation in Sect. 5.5. Conclusions and future work can be found in Sect. 5.6.

5.1 Previous work

While the problem of identifying the TFIs has received scant attention in the literature, a number of approaches have been proposed to filter the FIs of *spurious patterns*, i.e., patterns that are not actually *interesting* according to some measure of interestingness. We refer the reader to [Han et al., 2007, Sect. 3] and [Geng and Hamilton, 2006] for surveys on different measures. We remark that, as noted by Liu et al. [2011], that the use of the minimum support threshold θ, reflecting the level of domain significance, is complementary to the use of interestingness measures, and that "statistical significance measures and domain significance measures should be used together to filter uninteresting rules from different perspectives". The algorithm we present can be seen as a method to filter out patterns that are not interesting according to the measure represented by the true

frequency.

A number of works explored the idea to use statistical properties of the patterns in order to assess their interestingness. While this is not the focus of our work, some of the techniques and models proposed are relevant to our framework. Most of these works are focused on association rules, but some results can be applied to itemsets. In these works, the notion of interestingness is related to the deviation between the observed frequency of a pattern in the dataset and its expected support in a random dataset generated according to a well-defined probability distribution that can incorporate prior belief and that can be updated during the mining process to ensure that the most "surprising" patterns are extracted. In many previous works, the probability distribution was defined by a simple independence model: an item belongs to a transaction independently from other items [DuMouchel and Pregibon, 2001; Gionis et al., 2007; Hämäläinen, 2010; Kirsch et al., 2012; Megiddo and Srikant, 1998; Silverstein et al., 1998]. In contrast, our work does not impose any restriction on the probability distribution generating the dataset, with the result that our method is as general as possible.

Kirsch et al. [2012] developed a multi-hypothesis testing procedure to identify the best support threshold such that the number of itemsets with at least such support deviates significantly from its expectation in a random dataset of the same size and with the same frequency distribution for the individual items. In our work, the minimum threshold θ is an input parameter fixed by the user, and we identify a threshold $\hat{\theta} \geq \theta$ to guarantee that the collection of FIs w.r.t. $\hat{\theta}$ does not contain any false discovery.

Gionis et al. [2007] present a method to create random datasets that can act as samples from a distribution satisfying an assumed generative model. The main idea is to swap items in a given dataset while keeping the length of the transactions and the sum over the columns constant. This method is only applicable if one can actually derive a procedure to perform the swapping in such a way that the generated datasets are indeed random samples from the assumed distribution. For the problem we are interested in, such procedure is not available and indeed it would be difficult to obtain a procedure that is valid for any distribution, given our goal to develop a method that makes no assumption on the distribution. Considering the same generative model, Hanhijärvi [2011] presents a direct adjustment method to bound the probability of false discoveries by taking into consideration the actual number of hypotheses to be tested.

Webb [2007] proposes the use of established statistical techniques to control the probability of

false discoveries. In one of these methods (called holdout), the available data are split into two parts: one is used for pattern discovery, while the second is used to verify the significance of the discovered patterns, testing one statistical hypothesis at a time. A new method (layered critical values) to choose the critical values when using a direct adjustment technique to control the probability of false discoveries is presented by Webb [2008] and works by exploiting the itemset lattice. The method we present instead identify a threshold frequency such that all the itemsets with frequency above the threshold are TFIs. There is no need to test each itemset separately and no need to split the dataset.

Liu et al. [2011] conduct an experimental evaluation of direct corrections, holdout data, and random permutations methods to control the false positives. They test the methods on a very specific problem (association rules for binary classification).

In contrast with the methods presented in the works above, ours does not employ an explicit direct correction depending on the number of patterns considered as it is done in traditional multiple hypothesis testing settings. It instead uses the entire available data to obtain more accurate results,without the need to re-sampling it to generate random datasets or to split the dataset in two parts, being therefore more efficient computationally.

5.2 Preliminaries

In this section we introduce the definitions, lemmas, and tools that we will use throughout the work, providing the details that are needed in later sections.

Given a ground set $\mathcal{I}$ of *items*, let π be a probability distribution on $2^{\mathcal{I}}$. A *transaction* $\tau \subseteq \mathcal{I}$ is a single sample drawn from π. The *length* $|\tau|$ of a transaction τ is the number of items in τ. A *dataset* $\mathcal{D}$ is a bag of n transactions $\mathcal{D} = \{\tau_1, \ldots, \tau_n \ : \ \tau_i \subseteq \mathcal{I}\}$, i.e., of n *independent identically distributed* (i.i.d.) samples from π. We call a subset of $\mathcal{I}$ an *itemset*. For any itemset A, let $T(A) = \{\tau \subseteq \mathcal{I} \ : \ A \subseteq \tau\}$ be the *support set* of A. We define the *true frequency* $t_\pi(A)$ of A with respect to π as the probability that a transaction sampled from π contains A:

$$t_\pi(A) = \sum_{\tau \in T(A)} \pi(\tau) \ .$$

Analogously, given a (observed) dataset $\mathcal{D}$, let $T_{\mathcal{D}}(A)$ denote the set of transactions in $\mathcal{D}$ containing A. As we saw in previous chapters, the *frequency* of A in $\mathcal{D}$ is the fraction of transactions in

$\mathcal{D}$ that contain A: $f_{\mathcal{D}}(A) = |T_{\mathcal{D}}(A)|/|\mathcal{D}|$. It is easy to see that $f_{\mathcal{D}}(A)$ is the *empirical average* (and an *unbiased estimator*) for $t_\pi(A)$: $\mathbf{E}[f_{\mathcal{D}}(A)] = t_\pi(A)$.

In most applications the final goal of data mining is to gain a better understanding of the *process generating the data*, i.e., of the distribution π, through the true frequencies t_π, which are *unknown* and only approximately reflected in the dataset $\mathcal{D}$. Therefore, we are interested in finding the itemsets with *true* frequency t_π at least θ for some $\theta \in (0, 1]$. We call these itemsets the *True Frequent Itemsets* (TFIs) and denote their set as

$$\mathsf{TFI}(\pi, \mathcal{I}, \theta) = \{A \subseteq \mathcal{I} \ : \ t_\pi(A) \geq \theta\} \ .$$

If one is only given a *finite* number of random samples (the dataset $\mathcal{D}$) from π as it is usually the case, one can not aim at finding the exact set $\mathsf{TFI}(\pi, \mathcal{I}, \theta)$: no assumption can be made on the set-inclusion relationship between $\mathsf{TFI}(\pi, \mathcal{I}, \theta)$ and $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$, because an itemset $A \in \mathsf{TFI}(\pi, \mathcal{I}, \theta)$ may not appear in $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$, and vice versa. One can instead try to *approximate* the set of TFIs, which is what we aim at.

Goal. Given an user-specified parameter $\delta \in (0, 1)$, we aim at providing a threshold $\hat{\theta} \geq \theta$ such that $\mathcal{C} = \mathsf{FI}(\mathcal{D}, \mathcal{I}, \hat{\theta})$ *well approximates* $\mathsf{TFI}(\pi, \mathcal{I}, \theta)$, in the sense that

1. With probability at least $1 - \delta$, $\mathcal{C}$ does not contain any false positive:

$$\Pr(\exists A \in \mathcal{C} \ : \ t_\pi(A) < \theta) < \delta \ .$$

2. $\mathcal{C}$ contains as many TFIs as possible.

The method we present does not make *any* assumption about π. It uses information from $\mathcal{D}$, and guarantees a small probability of false positives while achieving a high success rate.

5.3 The range space of a collection of itemsets

In this section we define the concept of a range space associated to a collection of itemsets and show how to bound the VC-dimension and the empirical VC-dimension of this range space. We use these definitions and results to develop our algorithm in later sections.

Definition 12. Given a collection $\mathcal{C}$ of itemsets built on a ground set $\mathcal{I}$, the *range space* $(2^{\mathcal{I}}, \mathcal{R}(\mathcal{C}))$ *associated to* $\mathcal{C}$ is a range space on $2^{\mathcal{I}}$ such that $\mathcal{R}(\mathcal{C})$ contains the support sets of the itemsets in $\mathcal{C}$:

$$\mathcal{R}(\mathcal{C}) = \{T(A) \ : \ A \in \mathcal{C}\} \ .$$

This is a generalization of the range space from Def. 8. Given a dataset $\mathcal{D}$, Thm. 5 gives a bound to $\mathsf{EVC}(\mathcal{D}, \mathcal{R}(\mathcal{C}))$, the *empirical* VC-dimension of $\mathcal{R}(\mathcal{C})$ on $\mathcal{D}$ in the following sense.

Theorem 7. *Let $\mathcal{C}$ be a collection of itemsets and let $\mathcal{D}$ be a dataset. Let d be the maximum integer for which there are at least d transactions $\tau_1, \ldots, \tau_d \in \mathcal{D}$ such that the set $\{\tau_1, \ldots, \tau_d\}$ is an antichain, and each τ_i, $1 \leq i \leq d$, contains at least 2^{d-1} itemsets from $\mathcal{C}$. Then $\mathsf{EVC}(\mathcal{D}, \mathcal{R}(\mathcal{C})) \leq d$.*

5.3.1 Computing the VC-Dimension

The naïve computation of d according to the definition in Thm. 7 requires to scan the transactions one by one, compute the number of itemsets from $\mathcal{C}$ appearing in each transaction, and make sure to consider only itemsets constituting antichains. Given the very large number of transactions in typical dataset and the fact that the number of itemsets in a transaction is exponential in its length, this method would be computationally too expensive. An upper bound to d (and therefore to $\mathsf{EVC}(\mathcal{D}, \mathcal{R}(\mathcal{C}))$) can be computed by solving a *Set-Union Knapsack Problem* (SUKP) [Goldschmidt et al., 1994] associated to $\mathcal{C}$.

Definition 13 ([Goldschmidt et al., 1994])**.** Let $U = \{a_1, \ldots, a_\ell\}$ be a set of elements and let $\mathcal{S} = \{A_1, \ldots, A_k\}$ be a set of subsets of U, i.e. $A_i \subseteq U$ for $1 \leq i \leq k$. Each subset A_i, $1 \leq i \leq k$, has an associated non-negative *profit* $\rho(A_i) \in \mathbb{R}^+$, and each element a_j, $1 \leq j \leq \ell$ as an associated non-negative weight $w(a_j) \in \mathbb{R}^+$. Given a subset $\mathcal{S}' \subseteq \mathcal{S}$, we define the profit of $\mathcal{S}'$ as $P(\mathcal{S}') = \sum_{A_i \in \mathcal{S}'} \rho(A_i)$. Let $U_{\mathcal{S}'} = \cup_{A_i \in \mathcal{S}'} A_i$. We define the weight of $\mathcal{S}'$ as $W(\mathcal{S}') = \sum_{a_j \in U_{\mathcal{S}'}} w(a_j)$. Given a non-negative parameter c that we call *capacity*, the *Set-Union Knapsack Problem* (SUKP) requires to find the set $\mathcal{S}^* \subseteq \mathcal{S}$ which *maximizes* $P(\mathcal{S}')$ over all sets $\mathcal{S}'$ such that $W(\mathcal{S}') \leq c$.

In our case, U is the set of items that appear in the itemsets of $\mathcal{C}$, $\mathcal{S} = \mathcal{C}$, the profits and the weights are all unitary, and the capacity constraint is an integer ℓ. We call this optimization problem the *SUKP associated to $\mathcal{C}$ with capacity ℓ*. It is easy to see that the optimal profit of this SUKP is the maximum number of itemsets from $\mathcal{C}$ that a transaction of length ℓ can contain. In order to show how to use this fact to compute an upper bound to $\mathsf{EVC}(\mathcal{D}, \mathcal{R}(\mathcal{C}))$, we need to define some additional terminology. Let $\ell_1, \ldots, \ell_w$ be the sequence of the *transaction lengths* of $\mathcal{D}$, i.e., for each value ℓ for which there is at least a transaction in $\mathcal{D}$ of length ℓ, there is one (and only one) index i, $1 \leq i \leq w$

such that $\ell_i = \ell$. Assume that the ℓ_i's are labelled in decreasing order: $\ell_1 > \ell_2 > \cdots > \ell_w$. Let now L_i, $1 \leq i \leq w$ be the maximum number of transactions in $\mathcal{D}$ that have length at least ℓ_i and such that for no two τ', τ'' of them we have either $\tau' \subseteq \tau''$ or $\tau'' \subseteq \tau'$. Let now q_i be the optimal profit of the SUKP associated to $\mathcal{C}$ with capacity L_i, and let $b_i = \lfloor \log_2 q_i \rfloor + 1$. The sequences $(\ell_i)_1^w$ and a sequence $(L_i^*)^w$ of upper bounds to $(L_i)_1^w$ can be computed efficiently with a scan of the dataset. The following lemma uses these sequences to show how to obtain an upper bound to the empirical VC-dimension of $\mathcal{C}$ on $\mathcal{D}$.

Lemma 13. *Let j be the minimum integer for which $b_i \leq L_i$. Then $\mathsf{EVC}(\mathcal{D}, \mathcal{C}) \leq b_j$.*

Proof. If $b_j \leq L_j$, then there are at least b_j transactions which can contain $2^{b_j - 1}$ itemsets from $\mathcal{C}$ and this is the maximum b_i for which it happens, because the sequence $b_1, b_2, \ldots, b_w$ is sorted in decreasing order, given that the sequence $q_1, q_2, \ldots, q_w$ is. Then b_j satisfies the conditions of Lemma 7. Hence $\mathsf{EVC}(\mathcal{D}, \mathcal{C}) \leq b_j$. □ □

Corollary 2. *Let q be profit of the SUKP associated to $\mathcal{C}$ with capacity equal to $\ell = |\{a \in \mathcal{I} \ : \ \exists A \in \mathcal{C} \text{ s.t. } a \in A\}|$ (ℓ is the number of items such that there is at least one itemset in $\mathcal{C}$ containing them). Let $b = \lfloor \log_2 q \rfloor + 1$. Then $\mathsf{VC}(2^{\mathcal{I}}, \mathcal{R}(\mathcal{C})) \leq b$.*

Solving the SUKP optimally is NP-hard in the general case, although there are known restrictions for which it can be solved in polynomial time using dynamic programming [Goldschmidt et al., 1994]. Since we have unit weights and unit profits, our SUKP is equivalent to the *densest k-subhypergraph* problem, which can not be approximated within a factor of $2^{O(\log n)^{\delta}}$ for any $\delta > 0$ unless $3STA \in DTIME(2^{n^{3/4+\varepsilon}})$ [Hajiaghayi et al., 2006]. A greedy algorithm by Arulselvan [2014] allows a constant factor approximation if each items only appear in a constant fraction of itemsets of $\mathcal{C}$. For our case, it is actually *not necessary to compute the optimal solution* to the SUKP: any upper bound solution for which we can prove that there is no power of two between that solution and the optimal solution would result in the *same upper bound* to the (empirical) VC-dimension, while substantially speeding up the computation. This property can be specified in currently available optimization problem solvers (e.g., CPLEX), which can then can compute the bound to the (empirical) VC-dimension very fast even for very large instances with thousands of items and hundred of thousands of itemsets in $\mathcal{C}$, making this approach practical.

Refinements. It is possible to make some refinements to our computation of the *empirical* VC-dimension of a collection $\mathcal{C}$ of itemsets on a dataset $\mathcal{D}$. First of all, one can remove from $\mathcal{C}$ all itemsets that never appear in $\mathcal{D}$, as the corresponding ranges can not help shattering any set of

transactions in $\mathcal{D}$. Identifying which itemsets to remove requires a single linear scan of $\mathcal{D}$. Secondly, when computing the capacities L_i (i.e., their upper bounds L_i^*), we can ignore all the transactions that do not contain *any* of the itemsets in $\mathcal{C}$ (or the filtered version of $\mathcal{C}$), as there is no way of shatter them using the ranges corresponding to itemsets in $\mathcal{C}$. Both refinements aim at reducing the optimal value of the SUKP associated to $\mathcal{C}$, and therefore at computing a smaller bound to the empirical VC-dimension of $\mathcal{C}$ on $\mathcal{D}$.

The range space of all itemsets. The range space associated to $2^{\mathcal{I}}$ (seen as a collection of itemsets) is particularly interesting for us. It is possible to compute bounds to $\mathsf{VC}(2^{\mathcal{I}}, \mathcal{R}(2^{\mathcal{I}}))$ and $\mathsf{EVC}(\mathcal{D}, \mathcal{R}(2^{\mathcal{I}}))$ without having to solve a SUKP and using instead the d-index. Indeed, these are both consequences of Thm. 5.

Corollary 3. *Let $\mathcal{D}$ be a dataset with d-index d. Then* $\mathsf{EVC}(\mathcal{D}, \mathcal{R}(2^{\mathcal{I}})) \leq d$.

Corollary 4. $\mathsf{VC}(2^{\mathcal{I}}, \mathcal{R}(2^{\mathcal{I}})) = |\mathcal{I}| - 1$.

5.4 Finding the True Frequent Itemsets

In this section we present an algorithm that receives in input a dataset $\mathcal{D}$, a minimum frequency threshold θ, and a confidence parameter $\delta \in (0, 1)$ and identifies a threshold $\hat{\theta}$ such that, with probability at least δ, all itemsets with frequency at least $\hat{\theta}$ in $\mathcal{D}$ are True Frequent Itemsets with respect to θ. The threshold $\hat{\theta}$ can be used to find a collection $\mathcal{C} = \mathsf{FI}(\mathcal{D}, \mathcal{I}, \hat{\theta})$ of itemsets such that $\Pr(\exists A \in \mathcal{C} \text{ s.t. } t_\pi(A) < \theta) < \delta$.

The intuition behind the method is the following. Let $\mathcal{B}$ be the *negative border* of $\mathsf{TFI}(\pi, \mathcal{I}, \theta)$, that is the set of itemsets not in $\mathsf{TFI}(\pi, \mathcal{I}, \theta)$ but such that all their proper subsets are in $\mathsf{TFI}(\pi, \mathcal{I}, \theta)$. If we can find an ε such that $\mathcal{D}$ is an ε-approximation to $(2^{\mathcal{I}}, \mathcal{R}(\mathcal{B}), \pi)$ then we have that any itemset $A \in \mathcal{B}$ has a frequency $f_{\mathcal{D}}(A)$ in $\mathcal{D}$ less than $\hat{\theta} = \theta + \varepsilon$, given that it must be $t_\pi(A) < \theta$. By the antimonotonicity property of the frequency, the same holds for all itemsets that are supersets of those in $\mathcal{B}$. Hence, the only itemsets that can have frequency in $\mathcal{D}$ greater or equal to $\hat{\theta} = \theta + \varepsilon$ are those with true frequency at least θ. In the following paragraphs we show how to compute ε.

Let δ_1 and δ_2 be such that $(1 - \delta_1)(1 - \delta_2) \geq 1 - \delta$. Let $(2^{\mathcal{I}}, \mathcal{R}(2^{\mathcal{I}})$ be the range space of all itemsets. We use Corol. 4 (resp. Thm. 3) to compute an upper bound d' to $\mathsf{VC}\left((2^{\mathcal{I}}, \mathcal{R}(2^{\mathcal{I}}))\right)$ (resp. d'' to $\mathsf{EVC}(\mathcal{D}, \mathcal{R}(2^{\mathcal{I}}))$. Then we can use d' in Thm. 1 (resp. d'' in Thm. 2) to compute an ε_1' (resp. an

ε_1'') such that $\mathcal{D}$ is, with probability at least $1-\delta_1$, an ε_1'-approximation (resp. ε_1''-approximation) to $(2^{\mathcal{I}}, \mathcal{R}(2^{\mathcal{I}}), \pi)$.

Fact 1. *Let $\varepsilon_1 = \min\{\varepsilon_1', \varepsilon_1''\}$. With probability at least $1-\delta_1$, $\mathcal{D}$ is an ε_1-approximation to $(2^{\mathcal{I}}, \mathcal{R}(2^{\mathcal{I}}), \pi)$.*

We want to find an upper bound the (empirical) VC-dimension of $(2^{\mathcal{I}}, \mathcal{R}(\mathcal{B})$. To this end, we use the fact that the negative border of a collection of itemsets is a *maximal antichain* on $2^{\mathcal{I}}$. Let now $\mathcal{W}$ be the *negative border* of $\mathcal{C}_1 = \mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta - \varepsilon_1)$, $\mathcal{G} = \{A \subseteq \mathcal{I} \ : \ \theta - \varepsilon_1 \leq f_{\mathcal{D}}(A) < \theta + \varepsilon_1\}$, and $\mathcal{F} = \mathcal{G} \cup \mathcal{W}$.

Lemma 14. *Let $\mathcal{Y}$ be the set of maximal antichains in $\mathcal{F}$. If $\mathcal{D}$ is an ε_1-approximation to $(2^{\mathcal{I}}, \mathcal{R}(2^{\mathcal{I}}), \pi)$, then*

1. $\max_{\mathcal{A} \in \mathcal{Y}} \mathsf{EVC}(\mathcal{D}, \mathcal{R}(\mathcal{A})) \geq \mathsf{EVC}(\mathcal{D}, \mathcal{R}(\mathcal{B}))$, *and*
2. $\max_{\mathcal{A} \in \mathcal{Y}} \mathsf{VC}\left((2^{\mathcal{I}}, \mathcal{R}(\mathcal{A}))\right) \geq \mathsf{VC}\left((2^{\mathcal{I}}, \mathcal{R}(\mathcal{B}))\right)$.

Proof. Given that $\mathcal{D}$ is an ε_1-approximation to $(\mathcal{R}(2^{\mathcal{I}}), \pi)$, then $\mathsf{TFI}(\pi, \mathcal{I}, \theta) \subseteq \mathcal{G} \cup \mathcal{C}_1$. From this and the definition of negative border and of $\mathcal{F}$, we have that $\mathcal{B} \subseteq \mathcal{F}$. Since $\mathcal{B}$ is a maximal antichain, then $\mathcal{B} \in \mathcal{Y}$. Hence the thesis. □ □

In order to compute upper bounds to $\mathsf{VC}\left((2^{\mathcal{I}}, \mathcal{R}(\mathcal{B}))\right)$ and $\mathsf{EVC}(\mathcal{R}(\mathcal{B}), \mathcal{D})$ we can solve slightly modified SUKPs associated to $\mathcal{F}$ with the additional constraint that the optimal solution, which is a collection of itemsets, *must be a maximal antichain*. Lemma 13 still holds even for the solutions of these modified SUKPs. Using these bounds in Thms. 1 and 2, we compute an ε_2 such that, with probability at least $1-\delta_2$, $\mathcal{D}$ is an ε_2-approximation to $(\mathcal{R}(\mathcal{B}), \pi)$. Let

$$\hat{\theta} = \theta + \varepsilon_2 \ .$$

Theorem 8. *With probability at least $1-\delta$, $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \hat{\theta})$ contains no false positives:*

$$\Pr\left(\mathsf{FI}(\mathcal{D}, \mathcal{I}, \hat{\theta}) \subseteq \mathsf{TFI}(\pi, \mathcal{I}, \theta)\right) \geq 1 - \delta \ .$$

Proof. Consider the two events E_1="$\mathcal{D}$ is an ε_1-approximation for $(\mathcal{R}(2^{\mathcal{I}}), \pi)$" and E_2 ="$\mathcal{D}$ is an ε_2-approximation for $(\mathcal{R}(\mathcal{B}), \pi)$". From the above discussion and the definition of δ_1 and δ_2 it follows that the event $\mathsf{E} = \mathsf{E}_1 \cap \mathsf{E}_1$ occurs with probability at least $1-\delta$. Suppose from now on that indeed E occurs.

Since E_1 occurs, then Lemma 14 holds, and the bounds we compute by solving the modified SUKP problems are indeed bounds to $\mathsf{VC}\left((2^{\mathcal{I}}, \mathcal{R}(\mathcal{B}))\right)$ and $\mathsf{EVC}(\mathcal{R}(\mathcal{B}, \mathcal{D}))$. Since E_2 also occurs, then for any $A \in \mathcal{B}$ we have $|t_\pi(A) - f_{\mathcal{D}}(A)| \leq \varepsilon_2$, but given that $t_\pi(A) < \theta$ because the elements of $\mathcal{B}$ are not

TFIs, then we have $f_{\mathcal{D}}(A) < \theta + \varepsilon_2$. Because of the antimonotonicity property of the frequency and the definition of $\mathcal{B}$, this holds for any itemset that is not in $\mathsf{TFI}(\pi, \mathcal{I}, \theta)$. Hence, the only itemsets that can have a frequency in $\mathcal{D}$ at least $\hat{\theta} = \theta + \varepsilon_2$ are the TFIs, so $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \hat{\theta}) \subseteq \mathsf{TFI}(\pi, \mathcal{I}, \theta)$, which concludes our proof. □ □

The pseudocode of our method is presented in Alg. 2.

Exploiting additional knowledge about π. Our algorithm is completely *distribution-free*, i.e., it does not require any assumption about the unknown distribution π. On the other hand, when information about π is available, our method can exploit it to achieve better performances in terms of running time, practicality, and accuracy. For example, in most applications π will not generate any transaction longer than some known upper bound $\ell \ll |\mathcal{I}|$. Consider for example an online marketplace like Amazon: it is extremely unlikely (if not humanly impossible) that a single customer buys one of each available product. Indeed, given the hundred of thousands of items on sale, it is safe to assume that all the transactions will contains at most ℓ items, for some $\ell \ll |\mathcal{I}|$. Other times, like in an online survey, it is the nature of the process that limits the number of items in a transaction, in this case the number of questions. A different kind of information about the generative process may consists in knowing that some combination of items may never occur, because "forbidden" in some wide sense. Other examples are possible. All these pieces of information can be used to compute better (i.e., stricter) upper bounds to the VC-dimension $\mathsf{VC}\left((2^{\mathcal{I}}, \mathcal{R}(2^{\mathcal{I}}))\right)$. For example, if we know that π will never generate transactions with more than ℓ items, we can safely say that $\mathsf{VC}\left((2^{\mathcal{I}}, \mathcal{R}(2^{\mathcal{I}}))\right) \leq \ell$, a much stricter bound than $|\mathcal{I}| - 1$ from Corol. 4. This may result in a smaller ε_1, a smaller ε, and a smaller $\hat{\theta}$, which allows to produce more TFIs in the output collection. In the experimental evaluation, we show the positive impact of including additional information may on the performances of our algorithm.

5.5 Experimental evaluation

We conducted an extensive evaluation to assess the performances of the algorithm we propose. In particular, we used it to compute values $\hat{\theta}$ for a number of frequencies θ on different datasets, and compared the collection of FIs w.r.t. $\hat{\theta}$ with the collection of TFIs, measuring the number of false positives and the fraction of TFIs that were found.

Dataset	Freq. θ	TFIs	Times FPs	Times FNs
`accidents`	0.2	889883	100%	100%
`BMS-POS`	0.005	4240	100%	100%
`chess`	0.6	254944	100%	100%
`connect`	0.85	142127	100%	100%
`kosarak`	0.015	189	45%	55%
`pumsb*`	0.45	1913	5%	80%
`retail`	0.0075	277	10%	20%

Table 5.1: Fractions of times that $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ contained false positives and missed TFIs (false negatives) over 20 datasets from the same ground truth.

Implementation. We implemented the algorithm in Python 3.3. To mine the FIs, we used the C implementation by Grahne and Zhu [Grahne and Zhu, 2003]. Our solver of choice for the SUKPs was IBM® ILOG® CPLEX® Optimization Studio 12.3. We run the experiments on a number of machines with x86-64 processors running GNU/Linux 3.2.0.

Datasets generation. We evaluated the algorithm using pseudo-artificial datasets generated by taking the datasets from the FIMI'04 repository[1] as the *ground truth* for the true frequencies t_π of the itemsets. We considered the following datasets: `accidents`, `BMS-POS`, `chess`, `connect`, `kosarak`, `pumsb*`, and `retail`. These datasets differ in size, number of items, and, more importantly for our case, distribution of the frequencies of the itemsets [Goethals and Zaki, 2004]. We created a dataset by *sampling 20 million transactions uniformly at random* from a FIMI repository dataset. In this way the the true frequency of an itemset is its frequency in the original FIMI dataset. Given that our method to find the TFIs is distribution-free, this is a valid procedure to establish a ground truth. We used these enlarged datasets in our experiments, and use the original name of the datasets in the FIMI repository to annotate the results for the datasets we generated.

False positives and false negatives in $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$. In the first set of experiments we evaluated the performances, in terms of inclusion of false positives and false negatives in the output, of mining the dataset at frequency θ. Table 5.1 reports the fraction of times (over 20 datasets from the same ground truth) that the set $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ contained false positives (FP) and was missing TFIs (false negatives (FN)). In most cases, especially when there are many TFIs, the inclusion of false positives

[1] `http://fimi.ua.ac.be/data/`

when mining at frequency θ should be expected. This highlights a need for methods like the one presented in this work, as there is no guarantee that $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ only contains TFIs. On the other hand, the fact that some TFIs have frequency in the dataset *smaller* than θ (false negatives) points out how one can not aim to extract all and only the TFIs by using a fixed threshold approach (as the one we present).

Control of the false positives (Precision). In this set of experiments we evaluated how well the threshold $\hat{\theta}$ computed by our algorithm allows to avoid the inclusion of false negatives in $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \hat{\theta})$. To this end, we used a wide range of values for the minimum true frequency threshold θ (see Table 5.2) and fixed $\delta = 0.1$. We repeated each experiment on 20 different enlarged datasets generated from the same original FIMI dataset. In all the *hundreds* of runs of our algorithms, $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \hat{\theta})$ *never* contained *any false positive*, i.e., *always contained only TFIs*. In other words, the *precision* of the output was 1.0 in all our experiments. Not only our method can give a frequency threshold to extract only TFIs, but it is more *conservative*, in terms of including false positives, than what the theoretical analysis guarantees.

Inclusion of TFIs (Recall). In addition to avoid false positives in the results, one wants to include as many TFIs as possible in the output collection. To this end, we assessed what fraction of the total number of TFIs is reported in $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \hat{\theta})$. Since there were no false positives, this is corresponds to evaluating the *recall* of the output collection. We fixed $\delta = 0.1$, and considered different values for the minimum true frequency threshold θ (see Table 5.2). For each frequency threshold, we repeated the experiment on 20 different datasets sampled from the same original FIMI dataset, and found very small variance in the results. We compared the fraction of TFIs that our algorithm included in output with that included by the "Chernoff and Union bounds" (CU) method we presented in Introduction. We compared two variants of the algorithms: one ("vanilla") which makes no assumption on the generative distribution π, and another ("additional info") which assumes that the process will not generate any transaction longer than twice the longest transaction found in the original FIMI dataset. Both algorithms can be easily modified to include this information. In Table 5.2 we report the average fraction of TFIs contained in $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \hat{\theta})$. We can see that the amount of TFIs found by our algorithm is always very high: only a *minimal* fraction (often less than 3%) of TFIs do not appear in the output. This is explained by the fact that the value ε_2 computed in our

method (see Sect. 5.4) is always smaller than 10^{-4}. Moreover our solution *uniformly outperforms* the CU method, often by a huge margin, since our algorithm does not have to take into account all possible itemsets when computing $\hat{\theta}$. Only partial results are reported for the "vanilla" variant because of the very high number of items in the considered datasets: the mining of the dataset is performed at frequency threshold $\theta - \varepsilon_1$ and if there are many items, then the value of ε_1 becomes very high because the bound to the VC-dimension of $\mathcal{R}(2^{\mathcal{I}})$ is $|\mathcal{I}| - 1$, and as a consequence we have $\theta - \varepsilon_1 \leq 0$. We stress, though, that assuming no knowledge about the distribution π is not realistic, and usually additional information, especially regarding the length of the transactions, is available and can and should be used. The use of additional information gives flexibility to our method and improves its practicality. Moreover, in some cases, it allows to find an even larger fraction of the TFIs.

5.6 Conclusions

The usefulness of frequent itemset mining is often hindered by spurious discoveries, or false positives, in the results. In this work we developed an algorithm to compute a frequency threshold $\hat{\theta}$ such that the collection of FIs at frequency $\hat{\theta}$ is a good approximation the collection of True Frequent Itemsets. The threshold is such that that the probability of reporting *any* false positive is bounded by a user-specified quantity. We used tools from statistical learning theory and from optimization to develop and analyze the algorithm. The experimental evaluation shows that the method we propose can indeed be used to control the presence of false positives while, at the same time, extracting a very large fraction of the TFIs from huge datasets. There are a number of directions for further research. Among these, we find particularly interesting and challenging the extension of our method to other definitions of statistical significance for patterns and to other definitions of patterns such as sequential patterns [Low-Kam et al., 2013]. Also interesting is the derivation of better lower bounds to the VC-dimension of the range set of a collection of itemsets. Moreover, while this work focuses on itemsets mining, we believe that it can be extended and generalized to other settings of multiple hypothesis testing, and give another alternative to existing approaches for controlling the probability of false discoveries.

Algorithm 2: Compute freq. threshold $\hat{\theta}$ s. t. $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \hat{\theta})$ contains only TFIs with prob. at least $1 - \delta$.

Input : Dataset $\mathcal{D}$, freq. threshold $\theta \in (0, 1)$, confidence $\delta \in (0, 1)$

Output: Freq. threshold $\hat{\theta}$ s. t. $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \hat{\theta})$ contains only TFIs with prob. at least $1 - \delta$.

$\delta_1, \delta_2 \leftarrow 1 - \sqrt{1 - \delta}$ `// $\delta_1$ and $\delta_2$ do not need to have the same value`

$d'_1 \leftarrow$ upper bound to $\mathsf{VC}(\mathcal{R}(2^{\mathcal{I}}))$ (e.g., $|\mathcal{I}| - 1$)

$\varepsilon'_1 \leftarrow \sqrt{\frac{c}{|\mathcal{D}|}\left(d'_1 + \log \frac{1}{\delta_1}\right)}$

$d''_1 \leftarrow$ upper bound to $\mathsf{EVC}(\mathcal{R}(2^{\mathcal{I}}), \mathcal{D})$ `// i.e., d-index of` $\mathcal{D}$ `(see Ch. 3)`

$\varepsilon''_1 \leftarrow 2\sqrt{\frac{2d''_1 \log(|\mathcal{D}|+1)}{|\mathcal{D}|}} + \sqrt{\frac{2 \log \frac{2}{\delta}}{|\mathcal{D}|}}$

$\varepsilon_1 \leftarrow \min\{\varepsilon'_1, \varepsilon''_1\}$

$\mathcal{C}_1 = \mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta - \varepsilon_1)$

$\mathcal{G} = \{A \subseteq \mathcal{I} \ : \ \theta - \varepsilon_1 \leq f_{\mathcal{D}}(A) < \theta + \varepsilon_1\}$

$\mathcal{W} \leftarrow$ negative border of $\mathcal{C}_1$

$\mathcal{F} = \mathcal{G} \cup \mathcal{W}$

$U \leftarrow \{a \in \mathcal{I} : \exists A \in \mathcal{F} \text{ s.t. } a \in A\}$

$b'_2 \leftarrow$ `solveAntichainSUKP(`$U, \mathcal{F}, |U|$`)`

$d'_2 \leftarrow \lfloor \log_2 b'_2 \rfloor + 1$

$\varepsilon'_2 \leftarrow \sqrt{\frac{c}{|\mathcal{D}|}\left(d'_2 + \log \frac{2}{\delta_2}\right)}$

$\ell_1, \ldots, \ell_w \leftarrow$ transaction lengths of $\mathcal{D}$ `// see Sect. 5.3.1`

$L_1, \ldots, L_w \leftarrow$ the maximum number of transactions in $\mathcal{D}$ that have length at least ℓ_i, $1 \leq i \leq w$, and such that for no two τ', τ'' of them we have either $\tau' \subseteq \tau''$ or $\tau'' \subseteq \tau'$

$i \leftarrow 0$

while *True* **do**

  $b''_2 \leftarrow$ `solveAntichainSUKP(`$U, \mathcal{F}, \ell_i$`)`

  $d''_2 \leftarrow \lfloor \log_2 b''_2 \rfloor + 1$

  if $d''_2 < L_i$ **then**

    break

  else

    $i \leftarrow i + 1$

  end

end

$\varepsilon''_2 \leftarrow 2\sqrt{\frac{2d''_2 \log(|\mathcal{D}|+1)}{|\mathcal{D}|}} + \sqrt{\frac{2 \log \frac{2}{\delta}}{|\mathcal{D}|}}$

$\varepsilon_2 \leftarrow \min\{\varepsilon'_2, \varepsilon''_2\}$

return $\theta + \varepsilon_2$

			Reported TFIs (Average Fraction)			
			"Vanilla" (no info)		Additional Info	
Dataset	Freq. θ	TFIs	CU Method	This Work	CU Method	This Work
accidents	0.8	149	0.838	**0.981**	0.853	**0.981**
	0.7	529	0.925	**0.985**	0.935	**0.985**
	0.6	2074	0.967	**0.992**	0.973	**0.992**
	0.5	8057	0.946	**0.991**	0.955	**0.991**
	0.45	16123	0.948	**0.992**	0.955	**0.992**
	0.4	32528	0.949	0.991	0.957	**0.992**
	0.3	149545			0.957	**0.989**
	0.2	889883			0.957	**0.987**
BMS-POS	0.05	59	0.845	**0.938**	0.851	**0.938**
	0.03	134	0.879	**0.992**	0.895	**0.992**
	0.02	308	0.847	**0.956**	0.876	**0.956**
	0.01	1099	0.813	0.868	0.833	**0.872**
	0.0075	1896			0.826	**0.854**
	0.005	4240			0.762	**0.775**
chess	0.8	8227	0.964	**0.991**	0.964	**0.991**
	0.775	13264	0.957	**0.990**	0.957	**0.990**
	0.75	20993	0.957	**0.983**	0.957	**0.983**
	0.65	111239			0.972	**0.991**
	0.6	254944			0.970	**0.989**
connect	0.95	2201	0.802	**0.951**	0.802	**0.951**
	0.925	9015	0.881	**0.975**	0.881	**0.975**
	0.9	27127	0.893	**0.978**	0.893	**0.978**
	0.875	65959			0.899	**0.974**
	0.85	142127			0.918	**0.974**
kosarak	0.04	42	0.738	**0.939**	0.809	**0.939**
	0.035	50	0.720	**0.980**	0.780	**0.980**
	0.025	82			0.682	**0.963**
	0.02	121			0.650	**0.975**
	0.015	189			0.641	**0.933**
pumsb*	0.55	305	0.791	**0.926**	0.859	**0.926**
	0.5	679	0.929	**0.998**	0.957	**0.998**
	0.49	804	0.858	**0.984**	0.907	**0.984**
	0.475	1050			0.942	**0.996**
	0.45	1913			0.861	**0.976**
retail	0.03	32	0.625	**1.00**	0.906	**1.00**
	0.025	38	0.842	**0.973**	0.972	**0.973**
	0.0225	46	0.739	0.934	0.869	**0.935**
	0.02	55			0.882	**0.945**
	0.01	159			0.902	**0.931**
	0.0075	277			0.811	**0.843**

Table 5.2: Recall. Average fraction (over 20 runs) of reported TFIs in the output of an algorithm using Chernoff and Union bound and of the one presented in this work. For each algorithm we present two versions, one (Vanilla) which uses no information about the generative process, and one (Add. Info) in which we assume the knowlegde that the process will not generate any transaction longer than twice the size of the longest transaction in the original FIMI dataset. In bold, the best result (highest reported fraction).

Chapter 6

Approximating Betweenness Centrality

In this chapter we use VC-dimension to compute approximations of centrality indices of vertices in a graph. Centrality indices are fundamental metrics for network analysis. They express the relative importance of a vertex in the network. Some of them, e.g., degree centrality, reflect local properties of the underlying graph, while others, like betweenness centrality, give information about the global network structure, as they are based on shortest path computation and counting [Newman, 2010]. We are interested in *betweenness centrality* [Anthonisse, 1971; Freeman, 1977], that is, for every vertex in the graph, the fraction of shortest paths that goes through that vertex (see Section 6.2 for formal definitions) , and on some variants of it [Borgatti and Everett, 2006; Brandes, 2008; Opsahl et al., 2010]. Betweenness centrality has been used to analyze social and protein interaction networks, to evaluate traffic in communication networks, and to identify important intersections in road networks [Geisberger et al., 2008; Newman, 2010]. There exist polynomial-time algorithms to compute the exact betweenness centrality [Brandes, 2001], but they are not practical for the analysis of the very large networks that are of interest these days. Graphs representing online social networks, communication networks, and the web graph have millions of nodes and billions of edges, making a polynomial-time algorithm too expensive in practice. Given that data mining is exploratory in nature, approximate results are usually sufficient, especially if the approximation error is guaranteed to be within user-specified limits. In practice, the user is interested in the relative ranking of the

This chapter is an extended version of a work that originally appeared in the proceedings of WSDM'14 as [Riondato and Kornaropoulos, 2014].

vertices according to their betweenness, rather than the actual value of the betweenness, so a very good estimation of the value of each vertex is sufficiently informative for most purposes. It is therefore natural to develop algorithms that trade off accuracy for speed and efficiently compute high-quality approximations of the betweenness of the vertices. Nevertheless, in order for these algorithms to be practical it is extremely important that they scale well and have a low runtime dependency on the size of the network (number of vertices and/or edges).

Our contributions We present two randomized algorithms to approximate the betweenness centrality (and some of its variants) of the vertices of a graph. The first algorithm guarantees that the estimated betweenness values for all vertices are within an *additive* factor ε from the real values, with probability at least $1 - \delta$. The second algorithm focuses on the top-K vertices with highest betweenness and returns a *superset* of the top-K, while ensuring that the estimated betweenness for all returned vertices is within a *multiplicative* factor ε from the real value, with probability at least $1 - \delta$. This is the first algorithm to reach such a high-quality approximation for the set of top-K vertices. The algorithms are based on random sampling of shortest paths. The analysis to derive the sufficient sample size is novel and uses notions and results from VC-dimension theory. We define a range space associated with the problem at hand and prove strict bounds to its VC-dimension. The resulting sample size *does not depend on the size of the graph*, but only on the maximum number of vertices in a shortest path, a *characteristic quantity* of the graph that we call the *vertex-diameter*. For some networks, we show that the VC-dimension is actually at most a constant and so the sample size depends *only on the approximation parameters* and not on any property of the graph, a somewhat surprising fact that points out interesting insights. Thanks to the lower runtime dependency on the size of the network, our algorithms are *much faster and more scalable* than previous contributions [Brandes and Pich, 2007; Geisberger et al., 2008; Jacob et al., 2005], while offering the same approximation guarantees. Moreover, the amount of work performed by our algorithms per sample is also less than that of others algorithms. We extensively evaluated our methods on real graphs and compared their performances to the exact algorithm for betweenness centrality [Brandes, 2001] and to other sampling-based approximation algorithms [Brandes and Pich, 2007; Geisberger et al., 2008; Jacob et al., 2005], showing that our methods achieve a huge speedup (3 to 4 times faster) and scale much better as the number of vertices in the network grows.

Outline. We present related work in Sect. 6.1. Section 6.2 introduces the definitions and concepts that we use throughout the capter. A range space for the problem at hand and the bounds to its VC-dimension are presented in Sect. 6.3. Based on these results we develop and analyze algorithms for betweenness estimation that we present in Sect. 6.4. Extensions of our methods to various variants of the problem are presented in Sect. 6.5. Section 6.6 reports the methodology and the results of our extensive experimental evaluation.

6.1 Related work

Over the years, a number of centrality measures have been defined [Newman, 2010]. We focus on betweenness centrality and some of its variants.

Betweenness centrality was introduced in the sociology literature [Anthonisse, 1971; Freeman, 1977]. Brandes [2008] presents a number of minor variants. A particularly interesting one, k-bounded-distance betweenness, limits the length of the shortest paths considered when computing the centrality [Borgatti and Everett, 2006; Brandes, 2008; Pfeffer and Carley, 2012]. This is not to be confused with k-path betweenness centrality defined by Kourtellis et al. [2012], which considers simple random walks that are not necessarily shortest paths. Dolev et al. [2010] present a generalization of betweenness centrality which takes into account routing policies in the network. Opsahl et al. [2010] define a new distance function between pair of vertices in order to penalize paths with a high number of hops in a weighted network. This function induces a generalized and parametrized definition of betweenness.

The need of fast algorithms to compute the betweenness of vertices in a graph arose as large online social networks started to appear. Brandes [2001] presents the first efficient algorithm for the task, running in time $O(nm)$ on unweighted graphs and $O(nm+n^2 \log n)$ on weighted ones. The algorithm computes, for each vertex v, the shortest path to every other vertex and then traverses these paths backwards to efficiently compute the contribution of the shortest paths from v to the betweenness of other vertices. For very large networks, the cost of this algorithm would still be prohibitive in practice, so many approximation algorithms were developed [Bader et al., 2007; Brandes and Pich, 2007; Geisberger et al., 2008; Jacob et al., 2005; Lim et al., 2011; Maiya and Berger-Wolf, 2010]. The use of random sampling was one of the more natural approaches to speed up the computation of betweenness. Inspired by the work of Eppstein and Wang [2004], Jacob et al. [2005]

and independently Brandes and Pich [2007] present an algorithm that mimics the exact one, with the difference that, instead of computing the contribution of all vertices to the betweenness of the others, it only considers the contributions of some vertices sampled uniformly at random. To guarantee that all estimates are within ε from their real value with probability at least $1 - \delta$, the algorithm from [Brandes and Pich, 2007; Jacob et al., 2005] needs $O(\log(n/\delta)/\varepsilon^2)$ samples. The analysis for the derivation of the sample size uses Hoeffding bounds [Hoeffding, 1963] and the union bound [Mitzenmacher and Upfal, 2005]. Geisberger et al. [2008] noticed that this can lead to an overestimation of the betweenness of vertices that are close to the sampled ones and introduced different unbiased estimators that are experimentally shown to have smaller variance and do not suffer from this overshooting. Our algorithm is different from these because we sample, each time, a single random shortest path. This leads to a much smaller sample size and less work done for each sample, resulting in a much faster way to compute approximations of the betweenness with the same probabilistic guarantees. Although existing algorithms [Brandes and Pich, 2007; Geisberger et al., 2008; Jacob et al., 2005] can be extended to return a superset of the top-K vertices with highest betweenness, they only offer an *additive* approximation guarantee, while our algorithm for the top-K vertices offers a *multiplicative* factor guarantee, which is much stricter. We delve more in the comparisons with these algorithms in Sect. 6.4.3 and 6.6.

A number of works explored the use of adaptive sampling, in contrast with the previous algorithms (and ours) which use a fixed sample size. Bader et al. [2007] present an adaptive sampling algorithm which computes good estimations for the betweenness of high-centrality vertices, by keeping track of the partial contribution of each sampled vertex, obtained by performing a single-source shortest paths computation to all other vertices. Maiya and Berger-Wolf [2010] use concepts from expander graphs to select a connected sample of vertices. They estimate the betweenness from the sample, which includes the vertices with high centrality. They build the connected sample by adding the vertex which maximizes the number of connections with vertices not already in the sample. Modified versions of this algorithm and an extensive experimental evaluation appeared in [Lim et al., 2011]. The algorithm does not offer any guarantee on the quality of the approximations. Compared to these adaptive sampling approaches, our methods ensure that the betweenness of all (or top-K) vertices is well approximated, while using a fixed, predetermined amount of samples. Sarıyüce et al. [2013] present an algorithm that pre-processes the network in multiple ways by removing degree-1

vertices and identical vertices, and splits it in separate components where the computation of betweenness can be performed independently and then aggregated. They do not present an analysis of the complexity of the algorithm.

Kranakis et al. [1997] present a number of results on the VC-dimension of various range spaces for graphs (stars, connected sets of vertices, sets of edges), but do not deal with shortest paths. Abraham et al. [2011] use VC-dimension to speed up shortest path computation but their range space is different from the one we use: their ground set is the set of vertices while ours is defined on shortest paths.

6.2 Graphs and betweenness centrality

We now formally define the concepts we use in this chapter.

Let $G = (V, E)$ be a graph, where $E \subseteq V \times V$, with $n = |V|$ vertices and $m = |E|$ edges. The graph G can be directed or undirected. We assume that there are no self-loops from one vertex to itself and no multiple edges between a pair of vertices. Each edge $e \in E$ has a non-negative weight $\mathsf{w}(e)$. Given a pair of distinct vertices $(u, v) \in V \times V$, $u \neq v$, a *path* $p_{uv} \subseteq V$ *from* u *to* v is an ordered sequence of vertices $p_{uv} = (w_1, \ldots, w_{|p_{uv}|})$ such that $w_1 = u$, $w_{|p_{uv}|} = v$ and for each $1 \leq i < |p_{uv}|$, $(w_i, w_{i+1}) \in E$. The vertices u and v are called the *end points* of p_{uv} and the vertices in $\mathsf{Int}(p_{uv}) = p_{uv} \setminus \{u, v\}$ are the *internal vertices of* p_{uv}. The *weight* $\mathsf{w}(p_{uv})$ of a path $p_{uv} = (u = w_1, w_2, \cdots, w_{p_{|uv|}} = v)$ from u to v is the sum of the weights of the edges composing the path: $\mathsf{w}(p_{uv}) = \sum_{i=1}^{|p_{uv}|-1} \mathsf{w}((w_i, w_{i+1}))$. We denote with $|p_{uv}|$ the number of vertices composing the path and call this the *size of the path* p_{uv}. Note that if the weights are not all unitary, it is not necessarily true that $\mathsf{w}(p_{uv}) = |p_{uv}| - 1$. A special and degenerate path is the *empty path* $p_\emptyset = \emptyset$, which by definition has weight $\mathsf{w}(p_\emptyset) = \infty$, no end points, and $\mathsf{Int}(p_\emptyset) = \emptyset$.

Given two distinct vertices $(u, v) \in V \times V$, the *shortest path distance* d_{uv} between u and v is the weight of a path with minimum weight between u and v among all paths between u and v. If there is no path between u and v, $d_{uv} = \infty$. We call a path between u and v with weight d_{uv} a *shortest path between* u *and* v. There can be multiple shortest paths between u and v and we denote the set of these paths as $\mathcal{S}_{uv}$ and the number of these paths as $\sigma_{uv} = |\mathcal{S}_{uv}|$. If there is no path between u and v, then $\mathcal{S}_{uv} = \{p_\emptyset\}$[1]. We denote with $\mathbb{S}_G$ the union of all the $\mathcal{S}_{uv}$'s, for all pairs $(u, v) \in V \times V$

[1]Note that even if $p_\emptyset = \emptyset$, the set $\{p_\emptyset\}$ is not empty. It contains one element.

of distinct nodes $u \neq v$:

$$\mathbb{S}_G = \bigcup_{\substack{(u,v)\in V\times V \\ u\neq v}} \mathcal{S}_{uv} \ .$$

We now define a characteristic quantity of a graph that we will use throughout the paper.

Definition 14. Given a graph $G = (V, E)$, the *vertex-diameter* $\mathsf{VD}(G)$ *of* G is the size of the shortest path in G with maximum size:

$$\mathsf{VD}(G) = \max\{|p| \ : \ p \in \mathbb{S}_G\} \ .$$

The vertex-diameter is the maximum number of vertices among all shortest paths in G. If all the edge weights are unitary, then $\mathsf{VD}(G)$ is equal to $\mathsf{diam}(G) + 1$, where $\mathsf{diam}(G)$ is the number of edges composing the longest shortest path in G.

Given a vertex v, let $\mathcal{T}_v \subseteq \mathbb{S}_G$ be the set of all shortest paths that v is *internal* to:

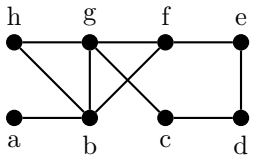
$$\mathcal{T}_v = \{p \in \mathbb{S}_G \ : \ v \in \mathsf{Int}(p)\} \ .$$

In this work we are interested in the *betweenness centrality* of the vertices of a graph.

Definition 15. [Anthonisse, 1971; Freeman, 1977] Given a graph $G = (V, E)$, the *betweenness centrality of a vertex* $v \in V$ is defined as[2]

$$\mathsf{b}(v) = \frac{1}{n(n-1)} \sum_{p_{uw}\in\mathbb{S}_G} \frac{\mathbb{1}_{\mathcal{T}_v}(p)}{\sigma_{uv}} \ .$$

Figure 6.1 shows an example of betweenness values for the vertices of a (undirected, unweighted) graph. It is intuitive, by looking at the graph, that vertices b and g should have higher betweenness than the others, given that they somehow act as bridges between two sides of the network, and indeed that is the case.



(a) Example graph

Vertex	a	b	c	d	e	f	g	h
$\mathsf{b}(v)$	0	0.250	0.125	0.036	0.054	0.080	0.268	0

(b) Betweenness values

Figure 6.1: Example of betweenness values

[2]We use the normalized version of betweenness as we believe it to be more suitable for presenting approximation results.

It is easy to see that $\mathsf{b}(v) \in [0,1]$. Brandes [2001] presented an algorithm to compute the betweenness centrality for all $v \in V$ in time $O(nm)$ for unweighted graphs and $O(nm + n^2 \log n)$ for weighted graphs.

We present many variants of betweenness in Sect. 6.5.

6.3 A range space of shortest paths

We now define a range space of the shortest paths of a graph $G = (V, E)$, and present a strict upper bound to its VC-dimension. We use the range space and the bound in the analysis of our algorithms for estimating the betweenness centrality of vertices of G.

The domain of the range space is the set $\mathbb{S}_G$ of all shortest paths between vertices of G. The set $\mathcal{R}_G$ of ranges contains a subset of $\mathbb{S}_G$ for each vertex $v \in V$, that is the set $\mathcal{T}_v$ of shortest paths that v is internal to:

$$\mathcal{R}_G = \{\mathcal{T}_v \ : \ v \in V\} \ .$$

We denote this range space as $\mathcal{R}_G$.

Lemma 15. $\mathsf{VC}\left((\mathbb{S}_G, \mathcal{R}_G)\right) \leq \lfloor \log_2(\mathsf{VD}(G) - 2) \rfloor + 1$.

Proof. Let $\ell > \lfloor \log_2(\mathsf{VD}(G)-2) \rfloor + 1$ and assume for the sake of contradiction that $\mathsf{VC}\left((\mathbb{S}_G, \mathcal{R}_G)\right) = \ell$. From the definition of the VC-dimension there is a set $Q \subseteq \mathbb{S}_G$ of size ℓ that is shattered by $\mathcal{R}_G$. Let p be an element of Q. There are $2^{\ell-1}$ non-empty subsets of Q containing the path p. Let us label these non-empty subsets of Q containing p as $S_1, \ldots, S_{2^{\ell-1}}$, where the labelling is arbitrary. Given that Q is shattered, for each set S_i there must be a range R_i in $\mathcal{R}_G$ such that $S_i = Q \cap R_i$. Since all the S_i's are different from each other, then all the R_i's must be different from each other. Given that p belongs to each S_i, then p must also belong to each R_i, that is, there are $2^{\ell-1}$ distinct ranges in $\mathcal{R}_G$ containing p. But p belongs only to the ranges corresponding to internal vertices of p, i.e., to vertices in $\mathsf{Int}(p)$. This means that the number of ranges in $\mathcal{R}_G$ that p belongs to is equal to $|p| - 2$. But $|p| \leq \mathsf{VD}(G)$, by definition of $\mathsf{VD}(G)$, so p can belong to at most $\mathsf{VD}(G) - 2$ ranges from $\mathcal{R}_G$. Given that $2^{\ell-1} > \mathsf{VD}(G) - 2$, we reached a contradiction and there cannot be $2^{\ell-1}$ distinct ranges containing p, hence not all the sets S_i can be expressed as $Q \cap R_i$ for some $R_i \in \mathcal{R}_G$. Then Q cannot be shattered and $\mathsf{VC}\left((\mathbb{S}_G, \mathcal{R}_G)\right) \leq \lfloor \log_2(\mathsf{VD}(G) - 2) \rfloor + 1$. □

Unique shortest paths. In the restricted case when the graph is undirected and every pair of distinct vertices has either none or a unique shortest path between them, the VC-dimension of $(\mathbb{S}_G, \mathcal{R}_G)$ reduces to a *constant*. This is a somewhat surprising result with interesting consequences. From a theoretical point of view, it suggests that there should be other characteristic quantities

of the graph different from the vertex diameter that control the VC-dimension of the range space of shortest paths, and these quantities are constant on graph with unique shortest paths between vertices. From a more practical point of view, we will see in Sect. 6.4 that this result has an impact on the sample size needed to approximate the betweenness centrality of networks where the unique-shortest-path property is satisfied or even enforced, like road networks [Geisberger et al., 2008]. In particular, the resulting sample size will be *completely independent* from any characteristic of the network, and will only be a function of the parameters controlling the desired approximation guarantees.

Lemma 16. *Let $G = (V, E)$ be an undirected graph with $|\mathcal{S}_{uv}| \leq 1$ for all pairs $(u, v) \in V \times V$. Then $\mathsf{VC}((\mathbb{S}_G, \mathcal{R}_G)) \leq 3$.*

Proof. First of all, notice that in this restricted setting, if two different shortest paths p_1 and p_2 meet at a vertex u, then they either go on together or they separate never to meet again at any other vertex $v \neq u$. This is easy to see: if they could separate at u and then meet again at some v, then there would be two distinct shortest paths between u and v, which is a contradiction of the hypothesis. Let us denote this fact as F.

Assume now that $\mathsf{VC}((\mathbb{S}_G, \mathcal{R}_G)) > 3$, then there must be a set $Q = \{p_1, p_2, p_3, p_4\}$ of four shortest paths that can be shattered by $\mathcal{R}_G$. Then there is a vertex w such that $\mathcal{T}_w \cap Q = Q$, i.e., all paths in Q go through w. Let x be the farthest predecessor of w along p_1 that p_1 shares with some other path from Q, and let y be the farthest successor of w along p_1 that p_1 shares with some other path from Q. It is easy to see that if either x or y (or both) do not exist, then Q cannot be shattered, as we would incur in a contradiction of fact F.

Let us then assume that both x and y exist. Let $Q_x = \mathcal{T}_x \cap Q$ and $Q_y = \mathcal{T}_y \cap Q$. Because of fact F, all paths in Q_x must go through the same vertices between x and w and all paths in Q_y must go through the same vertices between w and y. This also means that all paths in $Q_x \cap Q_y$ must go through the same vertices between x and y. If $Q_x \cup Q_y \neq Q$, let $p^* \in Q \setminus (Q_x \cup Q_y)$. Then from the definition of x and y and from fact F we have that there is no vertex v such that $\mathcal{T}_v \cap Q = \{p_1, p^*\}$, which implies that Q can not be shattered.

Suppose from now on that $Q_x \cup Q_y = Q$. If $Q_x \cap Q_y = Q$, then all the paths in Q go through the same vertices between x and y. From this and the definition of x and y we have that there is no vertex v such that, for example, $\mathcal{T}_v \cap Q = \{p_1, p_2\}$, hence Q cannot be shattered. Suppose instead that $Q_x \cap Q_y \neq Q$ and let $S = (Q_x \cap Q_y) \setminus \{p_1\}$. If $S \neq \emptyset$ then there is at least a path $p' \in S$ which, from the definition of S and fact F, must go through all the same vertices as p_1 between x and y. Moreover, given that $Q_x \cap Q_y \neq Q$, there must be a path $p^* \in Q \setminus \{p_1\}$ different from p_1 such that $p^* \notin S$. Then, from the definition of x, y, and S, and from the existence of p', there can be no vertex v such that $\mathcal{T}_v \cap Q = \{p_1, p^*\}$, hence Q cannot be shattered. Assume now that $S = \emptyset$ and consider the case $Q_x = \{p_1, p_2, p_3\}$, $Q_y = \{p_1, p_4\}$ (all other cases follow by symmetry with this case). Consider the set $\{p_1, p_3\}$. From the definition of x and Q_x, and from fact F we have that

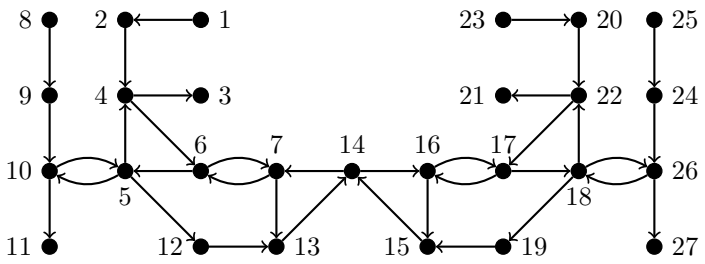


Figure 6.2: Directed graph $G = (V, E)$ with $|\mathcal{S}_{uv}| \leq 1$ for all pairs $(u, v) \in V \times V$ and such that it is possible to shatter a set of four paths.

there can not be a vertex v between the end point of p_1 before x and w such that $\mathcal{T}_v \cap Q = \{p_1, p_3\}$. At the same time, from the definition of y and from fact F, we have that such a v can not be between w and the end point of p_1 after y. This implies that Q can not be shattered.

We showed that in all possible cases we reached a contradiction and Q, which has size 4, can not be shattered by $\mathcal{R}_G$. Hence $\mathsf{VC}\left((\mathbb{S}_G, \mathcal{R}_G)\right) \leq 3$. □

It is natural to ask whether the above lemma or a similar result also holds for *directed* graphs. Fact F does not hold for directed graphs so the above proof does not extend immediately. In Fig. 6.2 we show a directed graph. For any pair of vertices in the graph, there is at most a unique shortest path connecting them. Consider now the following four directed shortest paths:

- $p_A = \{1, 2, 4, 6, 7, 13, 14, 16, 17, 18, 22, 21\}$
- $p_B = \{8, 9, 10, 5, 12, 13, 14, 16, 17, 18, 26, 27\}$
- $p_C = \{25, 24, 26, 18, 19, 15, 14, 7, 6, 5, 4, 3\}$
- $p_D = \{23, 20, 22, 17, 16, 15, 14, 7, 6, 5, 10, 11\}$

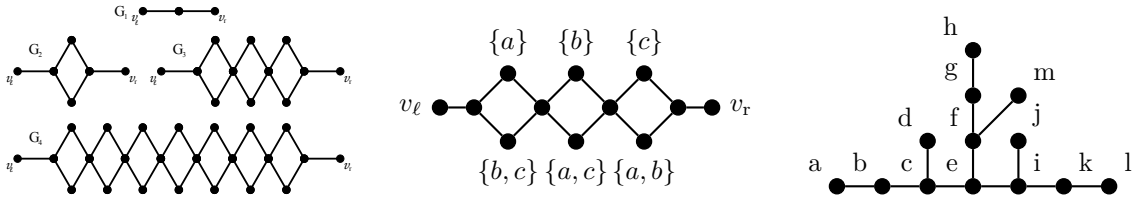
It is easy to check that the set $\{p_A, p_B, p_C, p_D\}$ is shattered, so the above lemma is not true for directed graphs. It is an open problem whether it is true for a different constant.

6.3.1 Tightness

The bound presented in Lemma 15 is strict in the sense that for each $d \geq 1$ we can build a graph G_d with vertex-diameter $\mathsf{VD}(G_d) = 2^d + 1$ and such that the range space $(\mathbb{S}_{G_d}, \mathcal{R}_{G_d})$ associated to the set of shortest paths of G_d has VC-dimension exactly $d = \lfloor \log_2(\mathsf{VD}(G_d) - 2) \rfloor + 1$.

We now introduce a class $\mathcal{G} = (G_d)_{d \geq 1}$ of graphs indexed by d. The graphs in $\mathcal{G}$ are the ones for which we can show the tightness of the bound to the VC-dimension of the associated range space.

We call the graph $G_d \in \mathcal{G}$ the d^{th} *concertina graph*. Figure 6.3a shows G_1, G_2, G_3, and G_4. The generalization to higher values of d is be straightforward. By construction, $\mathsf{VD}(G_d) = 2^d + 1$, so that $\lfloor \log_2(\mathsf{VD}(G_d) - 2) \rfloor + 1 = d$. The $3(2^{d-1})$ vertices of G_d can be partitioned into three classes, *top*, *bottom*, and *middle*, according to their location in a drawing of the graph similar to those in Fig. 6.3a. G_d has $2^{d-1} - 1$ top vertices, $2^{d-1} - 1$ bottom vertices, and $2^{d-1} + 2$ middle vertices. For each top vertex v, let $\mathsf{f}(v)$ be the *corresponding bottom vertex*, i.e., the bottom vertex u whose neighbors are the same middle vertices that are neighbors of v. Among the middle vertices, the two with degree 1 are special and are called the *end vertices* of G_d and denoted as v_ℓ and v_r, where the labels can be arbitrarily assigned. We now build a set Q of d shortest paths from v_ℓ to v_r and show that it is shattered by $\mathcal{R}_{G_d}$, therefore proving that $\mathsf{VC}\left((\mathbb{S}_{G_d}, R_{G_d})\right) \geq d$. This fact, together with Lemma 15, allows us to conclude that $\mathsf{VC}\left(\mathbb{S}_{G_d}, R_{G_d})\right) = d$.



(a) Examples of concertina graphs G_d for $d = 1, 2, 3, 4$.

(b) Example of the r map for G_3. The set next to each top and bottom vertex u is the set s s. t. $\mathsf{r}(s) = u$.

(c) Graph G with $\mathsf{VC}\left((\mathbb{S}_G, \mathcal{R}_G)\right) = 3$.

Figure 6.3: Graphs for which the bound is tight.

Lemma 17. $\mathsf{VC}\left((\mathbb{S}_{G_d} R_{G_d})\right) = d$.

Proof. As we said, the d paths in Q go from v_ℓ to v_r. From this and the definition of G_d it should be clear that they must go through all the middle vertices of G_d. Consider now the set $S = 2^Q \setminus \{Q, \emptyset\}$. We now build a map r from the elements of S to the set of top and bottom vertices of G_d. We can partition S in two sets A and B as follows: for each unordered pair (s', s'') of elements in S such that $s' \cap s'' = \emptyset$ and $s' \cup s'' = Q$ we put s' in A and s'' in B. It is easy to see that the size of A ($|A| = 2^{d-1} - 1$) equals the number of top vertices of G_d, and analogously for B and the number of bottom vertices. The bijection r will map the elements of A to the top vertices of G_d and the elements of B to the bottom vertices. For each $s' \in A$, let $\mathsf{c}(s')$ be the unique element s'' of B such that $s' \cap s'' = \emptyset$ and $s' \cup s'' = Q$ (i.e., $\mathsf{c}(s') = Q \setminus s'$). Let r_A be an arbitrary one-to-one map from the elements of A to the top vertices of G_d. Consider now the inverse map r_A^{-1} from the top vertices to the elements of A. We can create another map r_B from B to the bottom vertices of G_d that maps the element $\mathsf{c}(\mathsf{r}_A^{-1}(v))$ of B to the bottom vertex $\mathsf{f}(v)$ corresponding to v, for each top vertex v. A path $p \in Q$ goes through a top vertex v if and only if $p \in \mathsf{r}_A^{-1}(v)$. Analogously, p goes through a bottom vertex u if and only if $p \in \mathsf{r}_B^{-1}(u)$. It is easy to see that, if we combine r_A and r_B,

we obtain a map r from S to the set of top and bottom vertices of G_d. An example of a possible r for G_3 is presented in Fig. 6.3b. We now show that for each $s \in S$, $s = Q \cap \mathcal{T}_{\mathsf{r}(s)}$. This is easy to see as $\mathsf{r}(s)$ is internal to all paths in s, by definition of $\mathsf{r}(s)$ and of the paths in Q. On the other end, no path from $\mathsf{c}(s)$ goes through $\mathsf{r}(s)$ because it goes through the corresponding vertex of $\mathsf{r}(s)$ (top if $\mathsf{r}(s)$ is a bottom vertex, bottom otherwise). It is also straightforward to see that, if we let v_Q be any arbitrary middle vertex different from v_ℓ or v_r, we have $Q = Q \cap \mathcal{T}_{v_Q}$, given that all paths in Q go through all the middle vertices. Also, given that v_ℓ is not internal to any path, we have $\emptyset = Q \cap \mathcal{T}_{v_\ell}$. Then all subsets of Q can be expressed as the intersection between Q and a range from $\mathcal{R}_{G_d}$, which means that Q can be shattered and therefore $\mathsf{VC}\left((\mathbb{S}_G, \mathcal{R}_{G_d})\right) \geq d$.

From Lemma 15 we know that $\mathsf{VC}\left((\mathbb{S}_G, \mathcal{R}_{G_d})\right) \leq d$, so it must be $\mathsf{VC}\left((\mathbb{S}_G, \mathcal{R}_{G_d})\right) = d$. □

The upper bound presented in Lemma 16 for the case of unique shortest paths is also strict in the same sense.

Lemma 18. *There is a graph $G = (V, E)$ with $|\mathcal{S}_{uv}| \leq 1$ for all pairs $(u, v) \in V \times V$ such that the range space $(\mathbb{S}_G, \mathcal{R}_G)$ associated to the shortest paths in G has VC-Dimension exactly 3.*

Proof. Consider the graph G in Fig. 6.3c. Let $p_1 = (a, b, c, e, i, j)$, $p_2 = (m, f, e, i, k, l)$, $p_3 = (d, c, e, f, g, h)$ be three paths. We now show that $Q = \{p_1, p_2, p_3\}$ can be shattered by $\mathcal{R}_G$, which implies $\mathsf{VC}\left((\mathbb{S}_G, \mathcal{R}_G)\right) \geq 3$. We have $\emptyset = Q \cap \mathcal{T}_a$, $\{p_1\} = Q \cap \mathcal{T}_b$, $\{p_2\} = Q \cap \mathcal{T}_k$, $\{p_3\} = Q \cap \mathcal{T}_g$, $\{p_1, p_2\} = Q \cap \mathcal{T}_i$, $\{p_1, p_3\} = Q \cap \mathcal{T}_c$, $\{p_2, p_3\} = Q \cap \mathcal{T}_f$, $\{p_1, p_2, p_3\} = Q \cap \mathcal{T}_e$, Hence all subsets of Q can be expressed as the intersection between Q and some range in $\mathcal{R}_G$ which means that Q can be shattered and $\mathsf{VC}\left((\mathbb{S}_G, \mathcal{R}_G)\right) \geq 3$. Lemma 16 gives us an upper bound $\mathsf{VC}\left((\mathbb{S}_G, \mathcal{R}_G)\right) \leq 3$, so we can conclude that $\mathsf{VC}\left((\mathbb{S}_G, \mathcal{R}_G)\right) = 3$. □

Although the example in Fig. 6.3c is a tree, this is not a requirement for either Lemma 16 or Lemma 18: in a weighted graph with cycles (i.e., not a tree) the weights may be such that there is a unique shortest path between any pair of connected vertices.

6.4 Algorithms

In this section we present our algorithms to compute a set of approximations for the betweenness centrality of the (top-K) vertices in a graph through sampling, with probabilistic guarantees on the quality of the approximations.

6.4.1 Approximation for all the vertices

The intuition behind the algorithm to approximate the betweenness values of all vertices is the following. Given a graph $G = (V, E)$ with vertex-diameter $\mathsf{VD}(G)$ and two parameters $\varepsilon, \delta \in (0, 1)$

we first compute a sample size r using (2.2) with

$$d = \lfloor \log_2(\mathsf{VD}(G) - 2) \rfloor + 1 \ .$$

The resulting sample size is

$$r = \frac{c}{\varepsilon^2} \left(\lfloor \log_2(\mathsf{VD}(G) - 2) \rfloor + 1 + \ln \frac{1}{\delta} \right) \ . \tag{6.1}$$

This is sufficient to achieve the desired accuracy (expressed through ε) with the desired confidence (expressed through $1 - \delta$). The algorithm repeats the following steps r times: *1.* it samples a pair u, v of distinct vertices uniformly at random, *2.* it computes the set $\mathcal{S}_{uv}$ of all shortest paths between u and v, *3.* it selects a path p from $\mathcal{S}_{uv}$ uniformly at random, *4.* it increases by $1/r$ the betweenness estimation of each vertex in $\mathsf{Int}(p)$. Note that if the sampled vertices u and v are not connected, we can skip steps 3 and 4 because we defined $\mathcal{S}_{uv} = \{p_\emptyset\}$. Denoting with S the set of the sampled shortest paths, the *unbiased* estimator $\tilde{\mathsf{b}}(w)$ for the betweenness $\mathsf{b}(w)$ of a vertex w is the sample average

$$\tilde{\mathsf{b}}(w) = \frac{1}{r} \sum_{p \in S} \mathbb{1}_{\mathsf{Int}(p)}(w) = \frac{1}{r} \sum_{p \in S} \mathbb{1}_{\mathcal{T}_w}(p) \ .$$

There are two crucial steps in this algorithm: the computation of $\mathsf{VD}(G)$ and the sampling of a path uniformly at random from $\mathcal{S}_{uv}$. We first deal with the latter, and then present a linear-time constant-factor approximation algorithm for $\mathsf{VD}(G)$. Algorithm 3 presents the pseudocode of the algorithm, including the steps to select a random path. The `computeAllShortestPaths(`u, v`)` on line 8 is a call to a modified Dijkstra's (or BFS) algorithm to compute the set $\mathcal{S}_{uv}$, with the same modifications as [Brandes, 2001]. The `getDiameterApprox()` procedure computes an approximation for $\mathsf{VD}(G)$.

Unique shortest paths. When, for each pair (u, v) of vertices of G, either there is a unique shortest path from u to v or v is unreachable from u, then one can apply Lemma 16 and obtain a smaller sample size

$$r = \frac{c}{\varepsilon^2} \left(3 + \ln \frac{1}{\delta} \right)$$

to approximate the betweenness values of all the vertices. This is an interesting result: the number of samples needed to compute a good approximation to all vertices is a *constant* and completely *independent from* G. Intuitively, this means that the algorithm is extremely fast on graphs with

this property. Unique shortest paths are common or even enforced in road networks by slightly perturbing the edge weights or having a deterministic tie breaking policy [Geisberger et al., 2008].

Sampling a shortest path. Our procedure to select a random shortest path from $\mathcal{S}_{uv}$ is inspired by the dependencies accumulation procedure used in Brandes' exact algorithm [Brandes, 2001]. Let u and v be the vertices sampled by our algorithm (Step 7 of Alg. 3). We assume that u and v are connected otherwise the only possibility is to select the empty path $p_\emptyset$. Let y be any vertex belonging to at least one shortest path from u to v. Following Brandes [2001], we can compute σ_{uy} and $\mathcal{S}_{uy}$ while we compute the set $\mathcal{S}_{uv}$ of all the shortest paths from u to v. We can then use this information to select a shortest path p uniformly at random from $\mathcal{S}_{uv}$ as follows. For each vertex w let $P_u(w)$ be the subset of neighbors of w that are *predecessors* of w along the shortest paths from u to w. Let $p^* = \{v\}$. Starting from v, we select one of its predecessors $z \in P_u(v)$ using weighted random sampling: each $z \in P_u(v)$ has probability $\sigma_{uz} / \sum_{w \in P_u(v)} \sigma_{uw}$ of being sampled. We add z to p^* and then repeat the procedure for z. That is, we select one of z's predecessors from $P_u(z)$ using weighted sampling and add it to p^*, and so on until we reach u. Note that we can update the estimation of the betweenness of the internal vertices along p^* (the only ones for which the estimation is updated) as we compute p^*.

Lemma 19. *The path p^* built according to the above procedure is selected uniformly at random among the paths in $\mathcal{S}_{uv}$.*

Proof. The probability of sampling $p^* = (u, z_1, \ldots, z_{|p^*|-2}, v)$ equals to the product of the probabilities of sampling the vertices internal to p^*, hence

$$\Pr(p^*) = \frac{\sigma_{uz_{|p^*|-2}}}{\sigma_{uv}} \frac{\sigma_{uz_{|p^*|-3}}}{\sigma_{uz_{|p^*|-2}}} \cdots \frac{1}{\sigma_{uz_2}} = \frac{1}{\sigma_{uv}}$$

where we used [Brandes, 2001, Lemma3] which tells us that for $w \neq u$,

$$\sigma_{uw} = \sum_{j \in P_u(w)} \sigma_{uj}$$

and the fact that for z_1, which is a neighbor of u, $\sigma_{uz_1} = 1$. □

Approximating the vertex-diameter. The algorithm presented in the previous section requires the value of the vertex-diameter $\mathsf{VD}(G)$ of the graph G (line 4 of Alg. 3). Computing the exact value of $\mathsf{VD}(G)$ could be done by solving the All Pair Shortest Paths (APSP) problem, and taking

Algorithm 3: Computes approximations $\tilde{\mathsf{b}}(v)$ of the betweenness centrality $\mathsf{b}(v)$ for all vertices $v \in V$.

```
   Input  : Graph G = (V, E) with |V| = n, ε, δ ∈ (0, 1)
   Output: A set of approximations of the betweenness centrality of the vertices in V
 1 foreach w ∈ V do
 2 |  b̃(v) ← 0
 3 end
 4 VD(G) ←getVertexDiameter(G)
 5 r ← (c/ε²)(⌊log₂(VD(G) − 2)⌋ + ln(1/δ))
 6 for i ← 1 to r do
 7 |  (u, v) ←sampleUniformVertexPair(V)
 8 |  S_uv ←computeAllShortestPaths(u, v)
 9 |  if S_uv ≠ {p_∅} then
   |  |  //Random path sampling and estimation update
10 |  |  j ← v
11 |  |  s ← v
12 |  |  t ← v
13 |  |  while t ≠ u do
14 |  |  |  sample z ∈ P_s(t) with probability σ_uz/σ_us
15 |  |  |  if z ≠ u then
16 |  |  |  |  b̃(z) ← b̃(z) + 1/r
17 |  |  |  |  s ← t
18 |  |  |  |  t ← z
19 |  |  |  end
20 |  |  end
21 |  end
22 end
23 return {(v, b̃(v)), v ∈ V}
```

the shortest path with the maximum size. Algorithms for exactly solving APSP problem such as Johnson's which runs in $O(V^2 \log V + VE)$ or Floyd-Warshall's ($\Theta(V^3)$), would defeat our purposes: once we have all the shortest paths for the computation of the diameter, we could as well compute the betweenness of all the vertices exactly. Given that Thm. 1 (and Thm. 1) only requires an upper bound to the VC-dimension of the range set, an approximation of the vertex-diameter would be sufficient for our purposes. Several refined algorithms for approximating the diameter are known [Aingworth et al., 1999; Boitmanis et al., 2006; Roditty and Williams, 2012], with various running times and quality of approximations. We briefly present a well-known and simple approximation algorithm that has the right balance of accuracy and speed for our purposes.

Let $G = (V, E)$ be an *undirected* graph where *all the edge weights are equal*. It is a well-known result that one can obtain a 2-approximation $\widetilde{\mathsf{VD}}(G)$ of the vertex-diameter $\mathsf{VD}(G)$ of G in time $O(V + E)$ in the following way: 1. select a vertex $v \in V$ uniformly at random, 2. compute the

shortest paths from v to all other vertices in V, and 3. finally take $\widetilde{\mathsf{VD}}(G)$ to be the sum of the lengths of the two shortest paths with maximum size (which equals to the two longest shortest paths) from v to two distinct other nodes u and w. The approximation guarantee follows from the next lemma.

Lemma 20. $\mathsf{VD}(G) \leq \widetilde{\mathsf{VD}}(G) \leq 2\mathsf{VD}(G)$.

Proof. Let $v \in V$ be a vertex that we choose uniformly at random from the set V. Let also $u, w \in V$ be the two vertices such that the sum of the sizes of the shortest paths p_{vu} and p_{vw} is maximized among all the shortest paths that have v as a source. We have $\widetilde{\mathsf{VD}}(G) \leq 2\mathsf{VD}(G)$ because $|p_{vu}|, |p_{vw}| \leq \mathsf{VD}(G)$, so $|p_{vu}| + |p_{vw}| \leq 2\mathsf{VD}(G)$. To see that $\widetilde{\mathsf{VD}}(G) \geq \mathsf{VD}(G)$, consider a pair of vertices x and z such that the length of a shortest path between x and z is equal to $\mathsf{VD}(G)$. Let p_{xv} be a shortest path between x and v and let p_{vz} be a shortest path between v and z. From the properties of the shortest paths p_{vu}, p_{vw} we have $|p_{vu}| + |p_{vw}| \geq |p_{vx}| + |p_{vz}|$. Since the graph is undirected $|p_{s,t}| = |p_{t,s}|$ for every $s, t \in V$. Therefore:

$$\widetilde{\mathsf{VD}}(G) = |p_{vu}| + |p_{vw}| \geq |p_{vx}| + |p_{vz}| = |p_{xv}| + |p_{vz}| \geq \mathsf{VD}(G) \ .$$

For the last inequality we used the fact that since $\mathsf{VD}(G)$ is the size of the shortest path from x to z, then every other path (in this case p'_{xz} which is the merge of p_{xv} and p_{vz}) has greater or equal length from $p_{x,z}$. □

In case we have multiple connected components in G, we compute an upper bound to the vertex diameter of each component separately by running the above algorithm on each component, and then taking the maximum. The connected components can be computed in $O(n+m)$ by traversing the graph in a Breadth-First-Search (BFS) fashion starting from a random v. The time complexity of the approximation algorithm in the case of multiple connected components is again $O(n + m)$ since the sum of the vertices of individual components is n and the sum of edges is m.

The use of the above 2-approximation in the computation of the sample size from line 6 of Alg. 3 results in at most c/ε^2 additional samples than if we used the exact value $\mathsf{VD}(G)$. The computation of $\widetilde{\mathsf{VD}}(G)$ does not affect the running time of our algorithm: for the construction of the first sample we can reuse the shortest paths from the sampled vertex v that we used to obtain the approximation. Specifically, we can sample a new vertex $u \neq v$ and then choose with uniform probability one of the (already computed) shortest paths between v and u.

If the graph is directed and/or not all edge weights are equal, the computation of a good approximation to $\mathsf{VD}(G)$ becomes more problematic. In particular, notice that there is no relationship

between $\mathsf{VD}(G)$ and $\mathsf{diam}(G)$ when G is weighted, as the shortest path with maximum size may not be the shortest path with maximum weight. In these cases, one can use the size (number of vertices) of the largest Weakly Connected Component (WCC), as a loose upper bound to $\mathsf{VD}(G)$. The WCC's can again be computed in $O(n+m)$ using BFS. This quantity can be as high as n but for the computation of the sample size we use its logarithm, mitigating the crudeness of the bound. In this case our sample size is comparable to that proposed by Brandes and Pich [2007]. Nevertheless the amount of work done per sample by our algorithm is still much smaller (see Sect. 6.4.3 and 6.6 for more details). In practice, it is possible that the nature of the network suggests a much better upper bound to the vertex-diameter of the graph, resulting in a smaller sample size.

Analysis. Algorithm 3 offers probabilistic guarantees on the quality of all approximations of the betweenness centrality.

Lemma 21. *With probability at least $1-\delta$, all the approximations computed by the algorithm are within ε from their real value:*

$$\Pr\left(\exists v \in V \ s.t. \ |\mathsf{b}(v) - \tilde{\mathsf{b}}(v)| > \varepsilon\right) < \delta \ .$$

Proof. For each $p_{uv} \in \mathbb{S}_G$ let

$$\pi_G(p_{uv}) = \frac{1}{n(n-1)}\frac{1}{\sigma_{uv}} \ .$$

It is easy to see that π_G is a probability distribution and $\pi_G(p_{uv})$ is the probability of sampling the path p_{uv} during an execution of the loop on line 6 in Alg. 3, given the way that the vertices u and v are selected and Lemma 19.

Consider the range set $\mathcal{R}_G$ and the probability distribution π_G. Let S be the set of paths sampled during the execution of the algorithm. For r as in (6.1), Thm. 1 tells us that the sample S is a ε-approximation to $(\mathcal{R}_G, \pi_G)$ with probability at least $1-\delta$. Suppose that this is indeed the case, then from Def. 2 and the definition of $\mathcal{R}_G$ we have that

$$\left|\pi_G(\mathcal{T}_v) - \frac{1}{r}\sum_{p\in S}\mathbb{1}_{\mathcal{T}_v}(p)\right| = \left|\pi_G(\mathcal{T}_v) - \tilde{\mathsf{b}}(v)\right| \leq \varepsilon, \forall v \in V \ .$$

From the definition of π_G we have

$$\pi_G(\mathcal{T}_v) = \frac{1}{n(n-1)}\sum_{p_{uw}\in\mathcal{T}_v}\frac{1}{\sigma_{uw}} = \mathsf{b}(v),$$

which concludes the proof. □

Time and space complexity. Clearly the runtime of the algorithm is dominated by the computation of the shortest path at each step, which takes time $O(|V| + |M|)$ if the graph is unweighted (BFS algorithm) and time $O(|E| + |V| \log |V|)$ otherwise (Dijkstra's algorithm with Fibonacci heap). This time must then be multiplied by r as in (6.1) to obtain the final time complexity. The space requirements are dominated by the amount of memory needed to store the graph, so they are either $O(|V|^2)$ if using an adjacency matrix, or $O(|V| + |E|)$ if using $|V|$ adjacency lists.

6.4.2 High-quality approximation of the top-K betweenness vertices

Very often in practice one is interested only in identifying the vertices with the highest betweenness centrality, as they are the "primary actors" in the network. We present here an algorithm to compute a very high-quality approximation of the set $\mathsf{TOPK}(K, G)$ of the top-K betweenness vertices in a graph $G = (V, E)$. Formally, let $v_1, \ldots, v_n$ be a labelling of the vertices in V such that $\mathsf{b}(v_i) \geq \mathsf{b}(v_j)$ for $1 \leq i < j \leq n$. Then $\mathsf{TOPK}(K, G)$ is defined as the set of vertices with betweenness at least $\mathsf{b}(v_K)$:

$$\mathsf{TOPK}(K, G) = \{(v, \mathsf{b}(v)), \ : \ v \in V \text{ and } \mathsf{b}(v) \geq \mathsf{b}(v_K)\} \ .$$

Note that $\mathsf{TOPK}(K, G)$ may contain more than K vertices.

Our algorithm works in two phases. Each phase is basically a run of the algorithm for approximating the betweenness of all vertices. The two phases differ in the way they compute the number of paths to sample and the additional operations at the end of each phase. In the first phase, we compute a lower bound ℓ' to $\mathsf{b}(v_K)$. In the second phase we use ℓ' to compute the number of samples r needed to obtain a relative (ℓ', ε)-approximation to $(\mathcal{R}_G, \pi_G)$. We use r samples to approximate the betweenness of all vertices again, and return a collection of vertices that is, with high probability, a superset of $\mathsf{TOPK}(K, G)$.

Let $\widetilde{\mathsf{VD}}(G)$ be an upper bound to the vertex-diameter of G. Given $\varepsilon, \delta \in (0, 1)$, let δ', δ'' be two positive reals such that $(1 - \delta')(1 - \delta'') \geq (1 - \delta)$. Let

$$r' = \frac{c}{\varepsilon^2} \left(\lfloor \log_2(\widetilde{\mathsf{VD}}(G) - 2) \rfloor + 1 + \log \frac{1}{\delta'} \right) \ .$$

Let $\tilde{\mathsf{b}}'_k$ be the K-th highest estimated betweenness obtained using Algorithm 3 where $r = r'$, and

let $\ell' = \tilde{\mathsf{b}}_K - \varepsilon$, and

$$r'' = \frac{c'}{\varepsilon^2 \ell'} \left((\lfloor \log_2(\widetilde{\mathsf{VD}}(G) - 2) \rfloor + 1) \log \frac{1}{\ell'} + \log \frac{1}{\delta''} \right) \ .$$

We run Algorithm 3 with $r = r''$ and let $\tilde{\mathsf{b}}''_K$ be the so-obtained K-th highest estimated betweenness. Let $\ell'' = \min\{\tilde{\mathsf{b}}''(v)/(1+\varepsilon) \ : \ v \in V \text{ s.t. } \tilde{\mathsf{b}}''(v) \geq \tilde{\mathsf{b}}''_K\}$. We return the collection $\widetilde{\mathsf{TOPK}}(K, G)$ of vertices v such that $\tilde{\mathsf{b}}''(v) * (1+\varepsilon)/(1-\varepsilon) \geq \ell''$:

$$\widetilde{\mathsf{TOPK}}(K, G) = \left\{ v \in V \ : \ \tilde{\mathsf{b}}''(v) \frac{1+\varepsilon}{1-\varepsilon} \geq \ell'' \right\} \ .$$

The pseudocode of the algorithm is presented in Algorithm 4.

Algorithm 4: High-quality approximation of the top-K betweenness vertices

Input : a graph $G = (V, E)$ with $|V| = n$, a positive integer $K \leq n$, real values $\varepsilon, \delta \in (0, 1)$
Output: a superset of $\mathsf{TOPK}(K, G)$, with high-quality estimation of the betweenness for the vertices in the returned set.

$\delta', \delta'' \leftarrow$ two positive reals such that $(1-\delta')(1-\delta'') \geq (1-\delta)$
$\widetilde{\mathsf{VD}}(G) \leftarrow$ upper bound to $\mathsf{VD}(G)$
`//First phase`
$r' \leftarrow \frac{c}{\varepsilon^2} \left(\lfloor \log_2(\widetilde{\mathsf{VD}}(G) - 2) \rfloor + 1 + \log \frac{1}{\delta''} \right)$
$B' = \{(v, \tilde{\mathsf{b}}'(v)) \ : \ v \in V\} \leftarrow$ output of Algorithm 3 with $r = r'$
$\tilde{\mathsf{b}}'_K \leftarrow K$-th highest betweenness value from B', ties broken arbitrarily
$\ell' \leftarrow \tilde{\mathsf{b}}'_K - \varepsilon$
$r'' \leftarrow \frac{c'}{\varepsilon^2 \ell'} \left((\lfloor \log_2(\widetilde{\mathsf{VD}}(G) - 2) \rfloor + 1) \log \frac{1}{\ell'} + \log \frac{1}{\delta''} \right)$
`//Second phase`
$B'' = \{(v, \tilde{\mathsf{b}}''(v)) \ : \ v \in V\} \leftarrow$ output of Algorithm 3 with $r = r''$
$\tilde{\mathsf{b}}''_K \leftarrow K$-th highest betweenness value from B'', ties broken arbitrarily
$\ell'' \leftarrow \min\{\tilde{\mathsf{b}}''(v)/(1+\varepsilon) \ : \ v \text{ s.t. } \tilde{\mathsf{b}}''(v) \geq \tilde{\mathsf{b}}''_K\}$
return $\{(v, \tilde{\mathsf{b}}''(v)) \ : \ v \in V \text{ s.t. } \tilde{\mathsf{b}}''(v) \frac{1+\varepsilon}{1-\varepsilon} \geq \ell''\}$

Analysis. The following lemma shows the properties of the collection $\widetilde{\mathsf{TOPK}}(K, G)$.

Lemma 22. *With probability at least $1 - \delta$,*

1. $\mathsf{TOPK}(K, G) \subseteq \widetilde{\mathsf{TOPK}}(K, G)$, *and*
2. *for all $v \in \mathsf{TOPK}(K, G)$ we have $|\tilde{\mathsf{b}}''(v) - \mathsf{b}(v)| \leq \varepsilon \mathsf{b}(v)$, and*
3. *no vertex $u \in \widetilde{\mathsf{TOPK}}(K, G) \setminus \mathsf{TOPK}(K, G)$ has an estimated betweenness greater than $\ell'(1+\varepsilon)$.*

Proof. We start by proving 1. From Thm. 1 we know that, with probability at least $1-\delta'$, a sample of size r' is a ε-approximation to $(\mathcal{R}_G, \pi_G)$ and from Thm. 3 we have that with probability at least $1-\delta''$

a sample of size r'' is a relative (ℓ', ε)-approximation to $(\mathcal{R}_G, \pi_G)$. Suppose both these events occur, which happens with probability at least $1 - \delta$. Then it is easy to see that $\ell' \leq \mathsf{b}(v_K)$, as there must be at least K vertices with exact betweenness greater or equal to ℓ'. Consider now ℓ''. Following the same reasoning as for ℓ', it should be clear that $\ell'' \leq \mathsf{b}(v_K)$. The vertices included in $\widetilde{\mathsf{TOPK}}(K, G)$ are all and only those vertices that *may* have exact betweenness at least ℓ'', which implies that all vertices that have exact betweenness at least $\mathsf{b}(v_K)$ are included in $\widetilde{\mathsf{TOPK}}(K, G)$. Points 2 and 3 in the thesis follow from the properties of the relative (ℓ', ε)-approximation (Def. 3). □

The advantage of using our algorithm to approximate the collection of top-K betweenness vertices consists in the very high-quality approximation of the betweenness values for the returned set of vertices: they are all within a multiplicative factor ε from their exact values. By reverse-sorting them according to the approximated betweenness, one can obtain a ranking that is very similar to the original exact one. Previous algorithms could not achieve such a good ranking as they were only able to approximate the betweenness values to within an additive error ε. The cost of computing the high quality approximation for the top-K vertices is the cost of an additional run of our algorithm to compute good approximations for all the vertices.

6.4.3 Discussion

Jacob et al. [2005] and independently Brandes and Pich [2007] present a sampling-based algorithm to approximate the betweenness centrality of all the vertices of the graph. The algorithm (which we call BP) creates a sample $S = \{v_1, \ldots, v_r\}$ of r vertices drawn uniformly at random and computes all the shortest paths between each v_i to all other vertices in the graph. Their estimation $\tilde{\mathsf{b}}_{\mathsf{BP}}(u)$ for $\mathsf{b}(u)$ is

$$\tilde{\mathsf{b}}_{\mathsf{BP}}(u) = \frac{1}{(n-1)r} \sum_{v_i \in S} \sum_{\substack{w \neq v_i \\ w \neq u}} \sum_{p \in \mathcal{S}_{v_i w}} \frac{\mathbb{1}_{\mathsf{Int}(p)}(u)}{|\mathcal{S}_{v_i w}|} \ .$$

As it was for Algorithm 3, the key ingredient to ensure a correct approximation for the betweenness centrality is the computation of the sample size r. Inspired by the work of Eppstein and Wang [2004], Brandes and Pich [2007] prove that, to obtain good (within ε) estimations for the betweenness of all vertices with probability at least $1 - \delta$, it must be

$$r \geq \frac{1}{2\varepsilon^2} \left(\ln n + \ln 2 + \ln \frac{1}{\delta} \right) \ .$$

From this expression it should be clear that this sample size is usually much larger than ours, as in practice $\mathsf{VD}(G) \ll n$. For the same reason, this algorithm would not scale well as the network size

increases (see also Sect. 6.6).

Another interesting aspect in which our algorithm and BP differ is the *amount of work done per sample*. Our algorithm computes a single set $\mathcal{S}_{uv}$ for the sampled pair of vertices (u, v): it performs a run of Dijkstra's algorithm (or of BFS) from u, stopping when v is reached. BP instead computes all the sets $\mathcal{S}_{uw}$ from the sampled vertices u to all other vertices, again with a single run of Dijkstra or BFS, but without the "early-stopping condition" that we have when we reach v. Although in the worst case the two computations have the same time complexity[3], in practice we perform many fewer operations, as we can expect v not to always be very far from u and therefore we can terminate early. This fact has a huge impact on the running time. Our algorithm also touches *many fewer* edges than BP. The latter can touch all the edges in the graph *at every sample*, while our computation exhibits a much higher *locality*, exploring only a neighborhood of u until v is reached. The results of our experimental evaluation presented in Sect. 6.6 highlights this and other advantages of our method over the one from [Brandes and Pich, 2007; Jacob et al., 2005]. It would be interesting to explore the the possibility of using *bidirectional A^* search* [Kaindl and Kainz, 1997; Pohl, 1969] to further speed up the computation for each sample of our algorithms.

The analysis of Algorithm 1 allow us to obtain a tighter analysis for the algorithm by Brandes and Pich [2007] and Jacob et al. [2005].

Lemma 23. *Fix $r > 0$ and let $w \in V$. Let $\tilde{\mathsf{b}}_{\mathsf{BP}}(w)$ be the estimation of the betweenness $\mathsf{b}(w)$ computed by BP using r samples, and let $\tilde{\mathsf{b}}(w)$ be the estimation computed by Algorithm 3 using r samples. The estimator $\tilde{\mathsf{b}}_{\mathsf{BP}}(w)$ is* uniformly better *than the estimator $\tilde{\mathsf{b}}(w)$. I.e., for any value of $\mathsf{b}(w)$ and r, we have*

$$\mathrm{Var}[\tilde{\mathsf{b}}_{\mathsf{BP}}(w)] \leq \mathrm{Var}[\tilde{\mathsf{b}}(w)] \ .$$

Proof. Consider the quantity

$$\mathrm{Var}[\tilde{\mathsf{b}}(w)] - \mathrm{Var}[\tilde{\mathsf{b}}_{\mathsf{BP}}(w)] \ .$$

We will show that the above quantity is greater than or equal to 0, proving the thesis. Since $\mathbb{E}[\tilde{\mathsf{b}}(w)] = \mathbb{E}[\tilde{\mathsf{b}}_{\mathsf{BF}}(w)] = \mathsf{b}(w)$ and using the definition of variance, we only need to show

$$\mathbb{E}[(\tilde{\mathsf{b}}(w))^2] - \mathbb{E}[(\tilde{\mathsf{b}}_{\mathsf{BF}}(w))^2] \geq 0 \ . \tag{6.2}$$

[3]It is a well-known open problem whether there is an algorithm to perform a single s, t-shortest path computation between a pair of vertices with smaller worst-case time complexity than the Single Source Shortest Path computation.

Let start from computing $\mathbb{E}[(\tilde{\mathsf{b}}_{\mathsf{BF}}(w))^2]$. For every vertex v, we define α_v as

$$\alpha_v = \frac{1}{n-1} \sum_{\substack{u \in V \\ u \neq v}} \sum_{p \in \mathcal{S}_{vu}} \frac{\mathbb{1}_{\mathsf{Int}(p)}(w)}{\sigma_{vu}} \ .$$

Note that

$$\mathsf{b}(w) = \frac{1}{n} \sum_{v \in V} \alpha(v) \ . \tag{6.3}$$

Consider, for $1 \leq i \leq r$, the contribution X_i to $\tilde{\mathsf{b}}_{\mathsf{BP}}(w)$ of the paths computed from the i^{th} sampled vertex. X_i is a random variable that takes value α_v with probability $1/n$, for all $v \in V$. We have

$$\mathbb{E}[X_i] = \sum_{v \in V} \frac{1}{n} \alpha_v = \mathsf{b}(w) \text{ and } \mathbb{E}[X_i^2] = \frac{1}{n} \sum_{v \in V} \alpha_v^2, \forall 1 \leq i \leq r \ . \tag{6.4}$$

Clearly

$$\tilde{\mathsf{b}}_{\mathsf{BP}}(w) = \frac{1}{r} \sum_{i=1}^{r} X_i \ .$$

The variables X_i's are independent and identically distributed so

$$\begin{aligned} \mathbb{E}[(\tilde{\mathsf{b}}_{\mathsf{BP}}(w))^2] &= \frac{1}{r^2} \mathbb{E}\left[\left(\sum_{i=1}^{r} X_i \right)^2 \right] = \frac{1}{r^2} \sum_{i=1}^{r} \left(\mathbb{E}[X_i^2] + \sum_{j=i+1}^{r} (\mathbb{E}[X_i]\mathbb{E}[X_j]) \right) \\ &= \frac{1}{r} \frac{1}{n} \sum_{v \in V} \alpha_v^2 + \frac{r-1}{r} (\mathsf{b}(w))^2, \end{aligned} \tag{6.5}$$

where we used (6.4).

We now compute $\mathbb{E}[(\tilde{\mathsf{b}}(w))^2]$. Consider the contribution Y_i to $\tilde{\mathsf{b}}(w)$ of the i^{th} sampled path by Algorithm 1, for $1 \leq i \leq r$. Y_i is a random variable that takes value $\mathbb{1}_{\mathsf{Int}(p)}(w)$ with probability $\pi(p)$, for every shortest path $p \in \mathbb{S}_G$. We have

$$\mathbb{E}[Y_i] = \mathbb{E}[Y_i^2] = \mathsf{b}(w), \forall 1 \leq i \leq r \ . \tag{6.6}$$

By definition,

$$\tilde{\mathsf{b}}(w) = \frac{1}{r} \sum_{i=1}^{r} Y_i \ .$$

The random variables Y_i are independent and identically distributed so

$$\begin{aligned} \mathbb{E}[(\tilde{\mathsf{b}}(w))^2] &= \frac{1}{r^2} \mathbb{E}\left[\left(\sum_{i=1}^{r} Y_i \right)^2 \right] = \frac{1}{r^2} \sum_{i=1}^{r} \left(\mathbb{E}[Y_i^2] + \sum_{j=i+1}^{r} \mathbb{E}[Y_i]\mathbb{E}[Y_j] \right) \\ &= \frac{1}{r} \mathsf{b}(w) + \frac{r-1}{r} (\mathsf{b}(w))^2, \end{aligned} \tag{6.7}$$

where we used (6.6).

We can now rewrite the left side of (6.2) using (6.3), (6.5), and (6.7):

$$\mathbb{E}[(\tilde{\mathsf{b}}(w))^2] - \mathbb{E}[(\tilde{\mathsf{b}}_{\mathsf{BF}}(w))^2] = \frac{1}{r}\mathsf{b}(w) + \frac{r-1}{r}(\mathsf{b}(w))^2 - \frac{1}{r}\frac{1}{n}\sum_{v\in V}\alpha_v^2 - \frac{r-1}{r}(\mathsf{b}(w))^2$$
$$= \frac{1}{r}\frac{1}{n}\sum_{v\in V}(\alpha_v - \alpha_v^2) \ .$$

Since $\alpha_v \in [0,1]$, we have $\alpha_v - \alpha_v^2 \geq 0$ for all v, and our proof is complete. □

The following lemma is an easy consequence of the above.

Lemma 24. *BP has lower expected Mean Squared Error than Algorithm 1:*

$$\mathbb{E}[\mathrm{MSE}_{\mathsf{BP}}] \leq \mathbb{E}[\mathrm{MSE}] \ .$$

Proof. We have

$$\mathbb{E}[\mathrm{MSE}_{\mathsf{BP}}] = \mathbb{E}\left[\frac{1}{n}\sum_{v\in V}\left(\tilde{\mathsf{b}}_{\mathsf{BP}}(w) - \mathsf{b}(v)\right)^2\right] = \frac{1}{n}\sum_{v\in V}\mathbb{E}\left[\left(\tilde{\mathsf{b}}_{\mathsf{BP}}(w) - \mathsf{b}(v)\right)^2\right] = \frac{1}{n}\sum_{v\in V}\mathrm{Var}[\tilde{\mathsf{b}}_{\mathsf{BP}}(w)],$$

where we used the linearity of expectation and the fact that the estimator $\tilde{\mathsf{b}}_{\mathsf{BP}}(w)$ is unbiased for $\mathsf{b}(w)$ ($\mathbb{E}[\tilde{\mathsf{b}}_{\mathsf{BP}}(w)] = \mathsf{b}(w)$]). Analogously for Algorithm 1:

$$\mathbb{E}[\mathrm{MSE}] = \frac{1}{n}\sum_{v\in V}\mathrm{Var}[\tilde{\mathsf{b}}(w)] \ .$$

Hence

$$\mathbb{E}[\mathrm{MSE}] - \mathbb{E}[\mathrm{MSE}_{\mathsf{BP}}] = \frac{1}{n}\sum_{v\in V}\left(\mathrm{Var}[\tilde{\mathsf{b}}(w)] - \mathrm{Var}[\tilde{\mathsf{b}}_{\mathsf{BP}}(w)]\right),$$

and from Lemma 23 we have that each addend of the sum is non-negative and so is the above expression, concluding our proof. □

The estimator $\tilde{\mathsf{b}}_{\mathsf{BP}}(w)$ has lower variance and MSE than $\tilde{\mathsf{b}}(w)$, which means that it can give better estimations of the betweenness values in practice using the same number of samples. Before always opting for BP (or for the algorithm by Geisberger et al. [2008], whose estimators have even lower variance than those of BP) with a number of samples equal to the one in (6.1), one should nevertheless take into account two facts. Firstly, we do not currently have a proof that BP (or the algorithm from [Geisberger et al., 2008]) can compute, with a number of samples as in (6.1), a good (within $\pm\varepsilon$) approximation of the betweenness of all vertices with probability at least $1-\delta$. We conjecture this fact could be proven using pseudodimension [Anthony and Bartlett, 1999, Chap. 11]. Secondly we already argued that per sample, the computation of $\tilde{\mathsf{b}}_{\mathsf{BP}}(w)$ requires more time than

the one for $\tilde{\mathsf{b}}_{\mathrm{A1}}(w)$. The difference would be even larger if Algorithm 3 can be adapted to use bidirectional search [Kaindl and Kainz, 1997; Pohl, 1969].

We can conclude this discussion stating that Algorithm 3, BP, and the algorithm by Geisberger et al. [2008] share the same design principles, but choose different trade-offs between accuracy and speed.

6.5 Variants of betweenness centrality

In this section we discuss how to extends our results to some variants of betweenness centrality.

6.5.1 k-bounded-distance betweenness

A "local" variant of betweenness, called *k-bounded-distance betweenness*[4] only considers the contribution of shortest paths of size up to $k+1$ [Borgatti and Everett, 2006; Brandes, 2008]. For $k > 1$ and any pair of distinct vertices $u, v \in V$, $u \neq V$, let $\mathcal{S}_{uv}^{(k)} \subseteq \mathcal{S}_{uv}$ be the set of shortest paths from u to v of size at most $k+1$, with $\sigma_{uv}^{(k)} = |\mathcal{S}_{uv}^{(k)}|$, and let $\mathbb{S}_G^{(k)}$ be the union of all the $\mathcal{S}_{uv}^{(k)}$. Let $\mathcal{T}_v^{(k)} \subseteq \mathcal{T}_v$ be the set of all shortest paths *of size up to k* that v is internal to, for each $v \in V$.

Definition 16. [Borgatti and Everett, 2006; Brandes, 2008] Given a graph $G = (V, E)$ and an integer $k > 1$, the *k-bounded-distance betweenness centrality of a vertex $v \in V$* is defined as

$$\mathsf{bb}^{(k)}(v) = \frac{1}{n(n-1)} \sum_{p_{uw} \in \mathbb{S}_G^{(k)}} \frac{\mathbb{1}_{\mathcal{T}_v^{(k)}}(p)}{\sigma_{uw}^{(k)}} \ .$$

For the case of k-bounded-distance betweenness, if we let $\mathcal{R}_G^{(k)} = \{\mathcal{T}_v^{(k)} \ : \ v \in V\}$, it is easy to bound $\mathsf{VC}\left((\mathbb{S}_G^{(k)}, \mathcal{R}_G^{(k)})\right)$ following the same reasoning as in Lemma 15.

Lemma 25. $\mathsf{VC}\left((\mathbb{S}_G^{(k)}, \mathcal{R}_G^{(k)})\right) \leq \lfloor \log_2(k-1) \rfloor + 1$.

Given this result, the sample size on line 6 of Alg. 3 can be reduced to

$$r = \frac{c}{\varepsilon^2}\left(\lfloor \log_2(k-1) \rfloor + 1 + \ln \frac{1}{\delta}\right)$$

and the computation of the shortest paths on line 8 can be stopped after we reached the vertices that are k "hops" far from u.

[4] Bounded-distance betweenness is also known as k-betweenness. We prefer the former denomination to avoid confusion with k-path betweenness from [Kourtellis et al., 2012].

6.5.2 α-weighted betweenness

Opsahl et al. [2010] defined a parametric variant of betweenness centrality for weighted networks that can be seen as a generalization of the classical definition. The goal behind this new definition is to give the analyst the possibility of "penalizing" shortest paths with many vertices, given the intuition that there is a cost to be paid to have a message go through a vertex. The core of the new definition is a different weight function of a path $p_{uv} = (w_1 = u, w_2, \ldots, w_{|p_{uv}|} = v)$ between two vertices (u, v), parametrized by a real parameter $\alpha \geq 0$:

$$d_{\alpha,uv} = \sum_{i=1}^{|p_{uv}|-1} (\mathsf{w}((w_i, w_{i+1})))^{\alpha} \ . \tag{6.8}$$

Clearly for $\alpha = 0$, the weights are considered all equal to 1, and for $\alpha = 1$, the definition falls back to d_{uv}. For $0 < \alpha < 1$, paths with fewer vertices and higher weights are favored over longer paths with lighter edges[Opsahl et al., 2010]. The definition of shortest path distance, shortest path, and of betweenness centrality follow as before from (6.8). Dijkstra's shortest paths algorithm can be adapted to compute shortest paths according to the distance function in (6.8). To avoid confusion, we call the betweenness centrality computed using the shortest paths according to the distance function in (6.8), the *α-weighted betweenness* and denote it with $\mathsf{b}_{\alpha}(w)$.

For α-weighted betweenness we have that the VC-dimension of the range space defined on the shortest paths according to the distance from (6.8) is still bounded by $\lfloor \log_2(\mathsf{VD}(G)-1) \rfloor + 1$. Clearly, the vertex-diameter $\mathsf{VD}(G)$ is now defined as the maximum number of vertices in a shortest path according to (6.8).

No special arrangements are needed to obtain an approximation of α-weighted betweenness for all vertices. The sample size is the same as in (6.1).

6.5.3 k-path betweenness

Another interesting variant of betweenness centrality does not consider *shortest* paths, but rather *simple random walks* of size up to $k+1$, expressing the intuition that information in a network does not necessarily spread across shortest paths but has a high probability of "fading out" after having touched at most a fixed number of vertices.

Definition 17 ([Kourtellis et al., 2012])**.** Given a graph $G = (V, E)$ and a positive integer k, the

k-path centrality $\mathsf{pb}^{(k)}(v)$ of $v \in V$ is defined as the average[5], over all possible source vertices s, of the probability that a simple random walk originating from s and extinguishing after having touched (at most) $k+1$ vertices (s included) goes through v.

We can define another range space $(\mathbb{S}_G^{\mathrm{p},k}, \mathcal{R}_G^{\mathrm{p},k})$ if we are interested in k-path betweenness. The domain $\mathbb{S}_G^{\mathrm{p},k}$ of the range space now is the set of all simple random walks starting from any vertex of and of size up to $k+1$. For each vertex $v \in V$, the range R_v is the subset of $\mathbb{S}_G^{\mathrm{p},k}$ containing all and only the random walks from $\mathbb{S}_G^{\mathrm{p},k}$ that have v as internal vertex. It is easy to see that $\mathsf{VC}\left(\mathbb{S}_G^{\mathrm{p},k}, \mathcal{R}_G^{\mathrm{p},k})\right)$ is at most $\lfloor \log_2(k-1) + 1$ following the same reasoning as in Lemma 15 and Lemma 25.

For the case of k-path betweenness, the algorithm is slightly different: for

$$r = \frac{c}{\varepsilon^2}\left(\lfloor \log_2(k-1) \rfloor + 1 + \ln\frac{1}{\delta}\right)$$

iterations, rather than sampling a pair of vertices (uv), computing $\mathcal{S}_{uv}$, and then sampling a shortest path between them, we first sample a single vertex and an integer ℓ uniformly at random from $[1, k+1]$. We then sample a simple random walk touching ℓ vertices, while updating the estimated betweenness of each touched vertex by adding $1/r$ to its estimation. The method to sample a simple random walk is described in [Kourtellis et al., 2012].

Kourtellis et al. [2012] show that this algorithm can estimate, with probability at least $1 - 1/n^2$, all the k-path betweenness values to within an additive error $n^{-1/2+\alpha}/(n-1)$ using $2n^{1-2\alpha}k^2 \ln n$ samples, for $\alpha \in [-1/2, 1/2]$. We can obtain the same guarantees with a *much lower* number r of samples, precisely

$$r = 2n^{1-2\alpha}\left(\ln n + \frac{\lfloor \log_2(k-1) \rfloor + 1}{2}\right) \ .$$

In conclusion, we presented a tighter analysis of the algorithm by Kourtellis et al. [2012].

6.5.4 Edge betweenness

Until now we focused on computing the betweenness centrality of vertices. It is also possible to define a betweenness centrality index for *edges* of a graph $G = (V, E)$ [Anthonisse, 1971; Brandes, 2008]. Edge betweenness is useful, for example, to develop heuristics for community detection [Newman and Girvan, 2004]. Given $e \in E$, the edge betweenness $\mathsf{eb}(e)$ of e is defined as the fraction of shortest paths that *contain* e, meaning that if $e = (u, v)$, then a path p contains e if u and v appear

[5] We take the average, rather than the sum, as defined in the original work by Kourtellis et al. [2012], to normalize the value so that it belongs to the interval $[0, 1]$. This has no consequences on the results.

consecutively in p. Formally,

$$\mathsf{eb}(e) = \frac{1}{n(n-1)} \sum_{p_{uv} \in \mathbb{S}_G} \frac{\mathbb{1}_{p_{uv}}(e)}{\sigma_{uv}}, \forall e \in E \ .$$

We can then define a range space $(\mathbb{S}_G, \mathcal{ER}_G)$ that contains $|E|$ ranges R_e, one for each $e \in E$, where R_e is the set of shortest paths containing the edge e. By following the same reasoning as Lemma 15 we have that $\mathsf{VC}((\mathbb{S}_G, \mathcal{ER}_G)) \leq \lfloor (\log_2(\mathsf{VD}(G) - 1) \rfloor + 1$. Using this result we can adapt the sample sizes for Alg. 3 and 4 to compute good approximations of betweenness for the (top-k) edges.

				$\frac{\text{Time}_{\mathsf{BP}}}{\text{Time}_{\mathsf{VC}}}$	
	Graph Properties			diam-2approx	
Graph	$\lvert V\rvert$	$\lvert E\rvert$	$\mathsf{VD}(G)$	min	max
oregon1-010331	10,670	22,002	9	4.39	4.75
oregon1-010526	11,174	23,409	10	4.26	4.73
ca-HepPh	12,008	237,010	13	3.06	3.33
ca-AstroPh	18,772	396,160	14	3.26	3.76
ca-CondMat	23,133	186,936	15	3.75	4.08
email-Enron	36,692	421,578	12	3.60	4.16

(a) Undirected graphs

				$\frac{\text{Time}_{\mathsf{BP}}}{\text{Time}_{\mathsf{VC}}}$			
	Graph Properties			diam-exact		diam-UB	
Graph	$\lvert V\rvert$	$\lvert E\rvert$	$\mathsf{VD}(G)$	min	max	min	max
wiki-Vote	7,115	103,689	7	3.35	3.69	1.05	1.27
p2p-Gnutella25	22,687	54,705	11	5.45	5.78	1.94	2.09
cit-HepTh	27,770	352,807	14	3.58	3.83	1.39	1.61
cit-HepPh	34,546	421,578	12	4.91	5.01	1.60	1.71
p2p-Gnutella30	36,682	88,328	10	5.02	5.46	2.08	2.22
soc-Epinions1	75,879	508,837	13	4.20	4.25	1.35	1.38

(b) Directed graphs

Figure 6.4: Graph properties and running time ratios.

6.6 Experimental evaluation

We conducted an experimental evaluation of our algorithms, with two major driving goals in mind: study the behavior of the algorithms presented in this paper and compare it with that of other related algorithms [Brandes, 2001; Brandes and Pich, 2007; Geisberger et al., 2008; Jacob et al.,

2005], in terms of accuracy of the estimation, execution time, work performed, and scalability as function of the network size.

Implementation and environment. We implemented our algorithms, the one presented in [Brandes and Pich, 2007; Jacob et al., 2005] and the linear scaling version from [Geisberger et al., 2008] in C, by extending the implementation of the exact algorithm [Brandes, 2001] contained in igraph [Csárdi and Nepusz, 2006][6]. The implementations are similarly engineered, given that they are based on the same subroutines for the computation of the shortest path (Dijkstra's algorithm for weighted graphs, BFS for unweighted ones), and they received similar amounts of optimization. We exposed our implementations through Python 3.3.1, which was used for running the simulations. We run the experiments on a quad-core AMD Phenom™II X4 955 Processor with 16GB of RAM, running Debian *wheezy* with a Linux kernel version 3.2.0.

Datasets. In our evaluation we used a number of graphs from the Stanford Large Network Dataset Collection[7]. These are all real world datasets including online social networks, communication (email) networks, scientific citation and academic collaboration networks, road networks, Amazon frequent co-purchased product networks, and more. Basic information about the graphs we used are reported in the two leftmost columns of Figs. 6.4b and 6.4a. We refer the reader to the SLNDC website for additional details about each dataset. To evaluate scalability we also created a number of artificial graphs of different sizes (1,000 to 100,000 vertices) using the Barabási-Albert model [Barabási and Albert, 1999] as implemented by igraph [Csárdi and Nepusz, 2006].

Diameter approximation. As we discussed in the previous sections the number of samples that the proposed algorithm requires depends on the vertex-diameter of the graph. For the computation of the vertex-diameter in case of undirected graphs we used the 2-approximation algorithm that we briefly described in Sect. 6.4. We denote this as "diam-2-approx" when reporting results in this section. For directed graphs, we computed the number of samples using both the exact value of the vertex-diameter (indicated as diam-exact) as well as the trivial upper bound $|V| - 2$ (indicated as diam-UB).

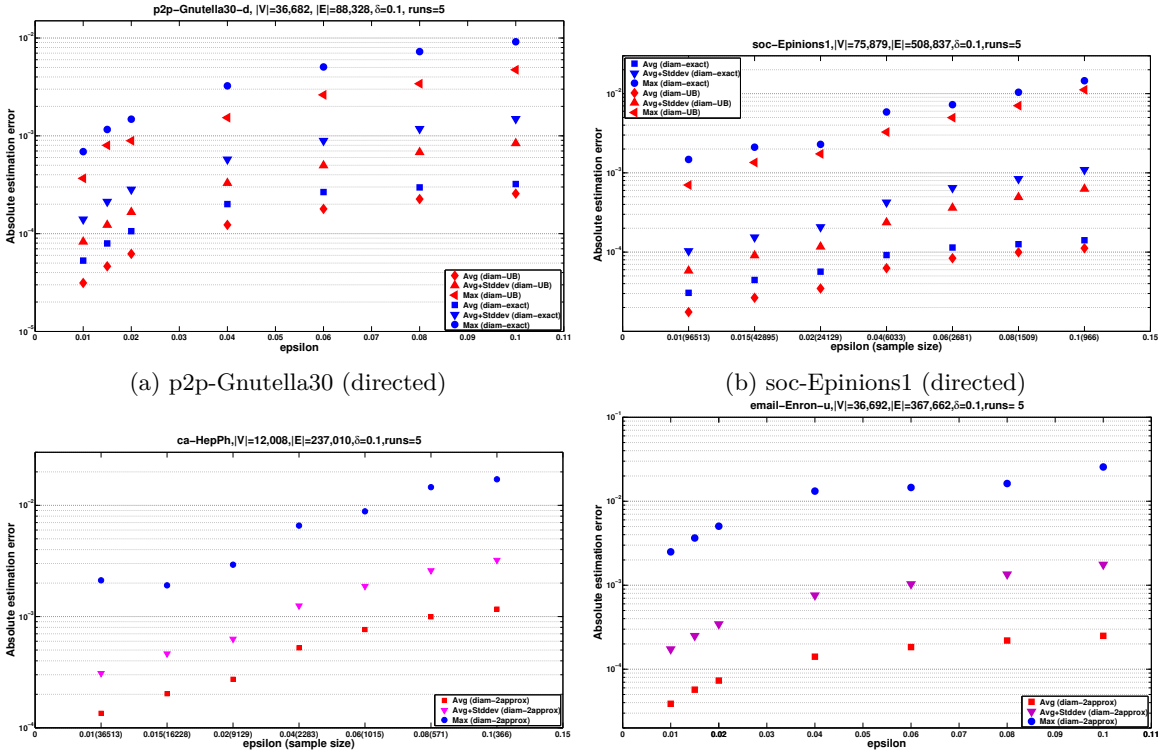
[6]The implementations are available at `http://cs.brown.edu/~matteo/centrsampl.tar.bz2`.
[7]`http://snap.stanford.edu/data/index.html`

6.6.1 Accuracy

Our theoretical results from Sect. 6.4 guarantee that, with probability at least $1-\delta$, all estimations of the betweenness values for all vertices in the graph are within ε for their real value. We run Algorithm 3 five times for each graph and each value of ε in $\{0.01, 0.015, 0.02, 0.04, 0.06, 0.08, 0.1\}$. The parameter δ was fixed to 0.1 and we used $c = 0.5$ in (2.2) to compute the sample size, as suggested by Löffler and Phillips [2009]. As far as the *confidence* is concerned, we report that in all the hundreds of runs we performed, the guarantee on the quality of approximation was *always* satisfied, not just with probability $1-\delta$ $(= 0.9)$. We evaluated how good the estimated values are by computing the average *estimation error* $(\sum_{v \in V} |\mathsf{b}(v) - \tilde{\mathsf{b}}(v)|)/|V|$ across five runs of our algorithm and taking the average and the standard deviation of this measure, for different values of ε. We also compute the maximum error $|\mathsf{b}(v) - \tilde{\mathsf{b}}(v)|$ overall. The results are reported in Fig. 6.5a for the directed graph p2p-Gnutella30, in Fig. 6.5c for the directed graph ca-HepPh, in Fig. 6.5b for the undirected graph soc-Epinions1, and in Fig. 6.5d for the undirected graph email-Enron. It is evident that the maximum error is an error of magnitude smaller than the guaranteed value of ε and that the average error is almost two orders of magnitude smaller than the guarantees, and the `Avg+Stddev` points show that the estimation are quite concentrated around the average. We can conclude that in practice the algorithm performs even *better than guaranteed*, achieving higher accuracy and confidence than what the theoretical analysis indicates. This is due to a number of factors, like the fact that we use an *upper bound* to the VC-dimension of the range set.

6.6.2 Runtime

We compared the running time of Algorithm 3 (denoted in the following as VC to that of the algorithm from [Brandes and Pich, 2007; Geisberger et al., 2008; Jacob et al., 2005] (denoted as BP), and to that of the exact algorithm [Brandes, 2001]. As VC and BP give the same guarantees on the accuracy and confidence of the computed estimations, it makes sense to compare their running times to evaluate which is faster in achieving the goal. The performances of the algorithm proposed in [Geisberger et al., 2008] takes the same time as BP, because it follows the same sampling approach and only differs in the definiton of the estimator for the betweenness, so we do not report those. The algorithms VC and BP take parameters ε and δ and compute the sample size accordingly. We run each experiments five times for each value of ε, and measured the average running time across the



(a) p2p-Gnutella30 (directed)

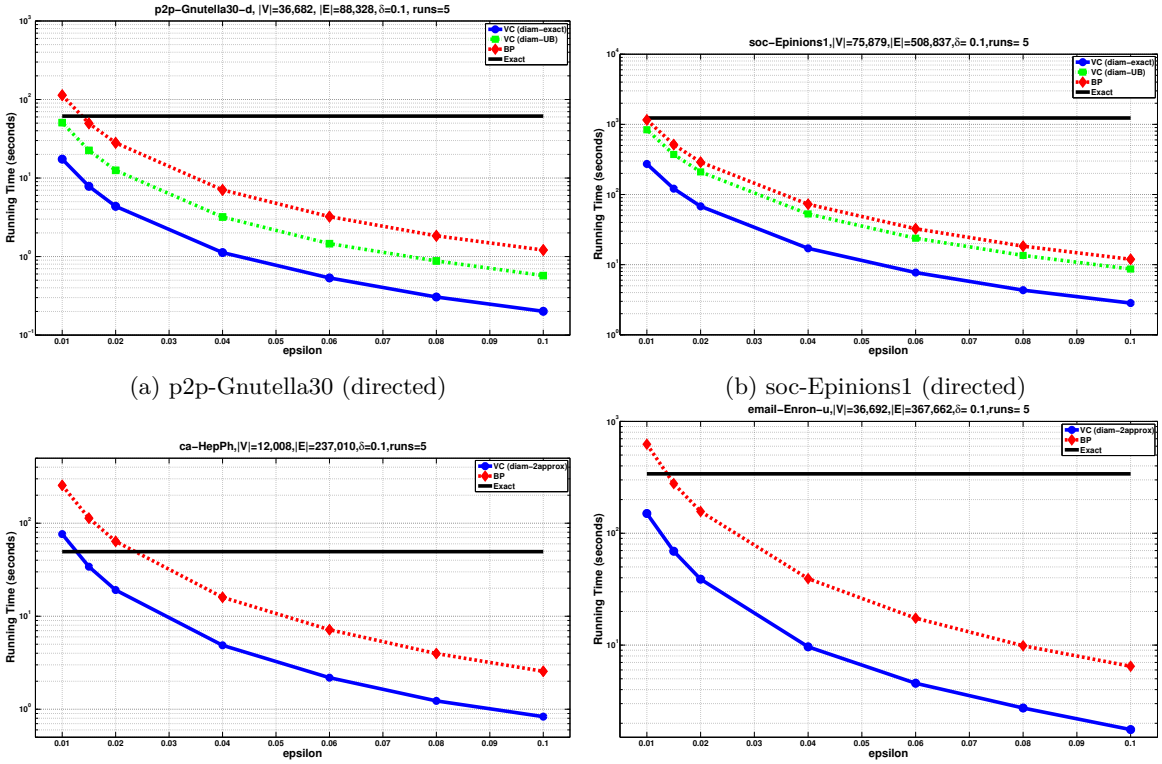
(b) soc-Epinions1 (directed)

(c) ca-HepPh (directed)

(d) email-Enron (undirected)

Figure 6.5: Betweenness estimation error $|\tilde{\mathsf{b}}(v) - \mathsf{b}(v)|$ evaluation for directed and undirected graphs

runs. The results are presented in Figs. 6.4 and 6.6. In Fig. 6.4a we report the minimum and the maximum ratio of the running time of BP over VC, taken over the ratios obtained by running the algorithms with the different values of ε. As it can be seen from this table our algorithm performs significantly faster, more than 300%. Similar results are reported for directed graphs in Fig. 6.4b. The diam-UB and the diam-exact values can be seen as the two extremes for the performance of Algorithm 3 in terms of runtime. In the case of the diam-exact we have as few samples as possible (for VC) since we use the exact value of the vertex-diameter, whereas in the case of diam-UB we have as many samples as possibles because we use the worst case estimation for the vertex-diameter of the graph. From Fig. 6.4a we can see that the value for the vertex-diameter that we consider in the case of diam-UB ($|V| - 2$) is many orders of magnitudes greater than the actual value, which translates in a significant increase of the number of samples. But even in the case of this crude vertex-diameter approximation (diam-UB), the VC algorithm performs uniformly faster than BP. In the case where the exact value of the diameter was used, we can see that our algorithm computes an estimation of the betweenness that satisfies the desired accuracy and confidence guarantees *3 to 5*



(a) p2p-Gnutella30 (directed)

(b) soc-Epinions1 (directed)

(c) ca-HepPh (directed)

(d) email-Enron (undirected)

Figure 6.6: Running time (seconds) comparison between VC, BP, and the exact algorithm.

times faster than BP. In Fig. 6.6a we study the directed graph p2p-Gnutella30 and we present the measurements of the average running time of the algorithms for different values of ε, using the exact algorithm from [Brandes, 2001] as baseline. The VC algorithm requires significantly less time than the BP algorithm. The figure also shows that there are values of ε for which BP takes more time than the exact algorithm, because the resulting sample size is larger than the graph size. Given that VC uses fewer samples and does fewer operations per sample, it can be used with lower ε than BP, while still saving time compared to the exact computation. Figure 6.6d shows the average running time of the algorithms for the undirected graph email-Enron. The behavior is similar to that for the undirected case. Algorithm 3 is faster than BP for two reasons, both originating from from our use of results from the VC-dimension theory: 1) we use a significantly smaller amount of samples and 2) VC performs the same amount of computations *per sample* as BP only in the worst case. Indeed our algorithm needs only to find the shortest path between a sampled pair of vertices, whereas the algorithms from [Brandes and Pich, 2007; Geisberger et al., 2008] need to compute the shortest paths between a sampled source and all the other vertices. In our experimental evaluation we found

out that the running time of the algorithms is directly proportional to the number of edges touched during the shortest path computation. The use of bidirectional A* search [Kaindl and Kainz, 1997; Pohl, 1969] can help in lowering the number of touched edges for VC and therefore the runtime of our algorithm (BP would not benefit from this improvement). This is a possible direction for future research.

6.6.3 Scalability

In Sect. 6.4.3 we argued about the reasons why Algorithm 3 is more scalable than BP, while still offering the same approximation guarantees. To evaluate our argument in practice, we created a number of graphs of increasing size (1,000 to 100,000 vertices) using the Barabási-Albert [Barabási and Albert, 1999] and run the algorithms on them, measuring their running time. We report the results in Fig. 6.7. The most-scalable algorithm would be completely independent from the size (number of vertices) of the graph, corresponding to a flat (horizontal) line in the plot. Therefore, the less steep the line, the more independent from the network size would be the corresponding algorithm. From the figure, we can appreciate that this is the case for VC, which is much more scalable and independent from the size of the sample than BP. This is very important, as today's networks are not only huge, but they also grow rapidly, and algorithms to mine them must scale well with graph size.

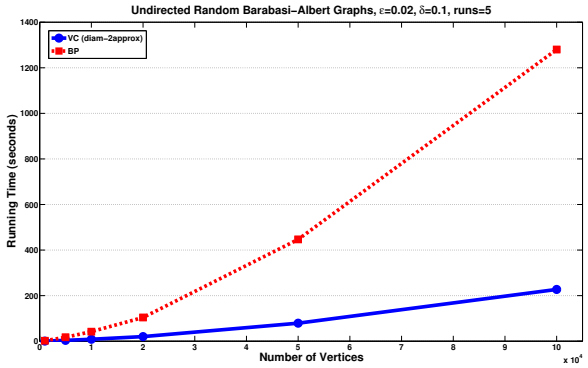


Figure 6.7: Scalability on random Barabási-Albert [Barabási and Albert, 1999] graphs.

6.7 Conclusions

In this chapter we presented two random-sampling-based algorithms for accurately and efficiently estimate the betweenness centrality of the (top-K) vertices in a graph, with high probability. Our

algorithms are based on a novel application of VC-dimension theory, and therefore take a different approach than previous ones achieving the same guarantees [Brandes and Pich, 2007; Geisberger et al., 2008; Jacob et al., 2005]. The number of samples needed to approximate the betweenness with the desired accuracy and confidence does not depend on the number of vertices in the graph, but rather on a characteristic quantity of the network that we call *vertex-diameter*. In some cases, the sample size is completely independent from any property of the graph. Our methods can be applied to many variants of betweenness, including edge betweenness. Our algorithms perform much less work than previously presented methods. As a consequence, they are much faster and scalable, as verified in the extensive experimental evaluation using many real and artificial graphs.

Chapter 7

Estimating the selectivity of SQL queries

In this chapter we examine how it is possible to use VC-dimension to compute a good sample of a database that can be used for estimating the selectivity of SQL queries.

As advances in technology allow for the collection and storage of vast databases, there is a growing need for *advanced machine learning techniques* for speeding up the execution of queries on such large datasets. In this chapter we focus on the fundamental task of estimating the selectivity, or output size, of a database query, which is a crucial step in a number of query processing tasks such as execution plan optimization and resource allocation in parallel and distributed databases. The task of efficiently obtaining such accurate estimates has been extensively studied in previous work with solutions ranging from storage of pre-computed statistics on the distribution of values in the tables, to online sampling of the databases, and to combinations of the two approaches [Ganguly et al., 1996; Ganti et al., 2000; Gibbons and Matias, 1998; Haas and Swami, 1992, 1995; Hou et al., 1988, 1991; Larson et al., 2007; Lipton and Naughton, 1995; Lipton et al., 1990; Poosala and Ioannidis, 1997]. Histograms, simple yet powerful statistics of the data in the tables, are the most commonly used solution in practice, thanks to their computational and space efficiency. However, there is an inherent limitation to the accuracy of this approach when estimating the selectivity of queries that involve either multiple tables/columns or correlated data. Running the query on freshly sampled data gives more accurate estimates at the cost of delaying the execution of the query while collecting

This chapter is an extended version of a work that originally appeared in the proceedings of ECML PKDD 2011 [Riondato et al., 2011].

random samples from a disk or other large storage medium and then performing the analysis itself. This approach is therefore usually more expensive than a histogram lookup. Our goal is to exploit both the computational efficiency of using pre-collected data and the provable accuracy of estimates obtained by running a query on a properly sized random sample of the database.

We apply VC-dimension to develop and analyze a novel technique to generate accurate estimates of query selectivity. A major theoretical contribution of this work, which is of independent interest, is an explicit bound to the VC-dimension of various classes of queries, viewed as indicator functions on the Cartesian product of the database tables. In particular, we show an upper bound to the VC-dimension of a class of queries that is a function of the maximum number of Boolean, select and join operations in any query in the class, but it is not a function of the number of different queries in the class. By adapting a fundamental result from the VC-dimension theory to the database setting, we develop a method that for any family of queries, defined by its VC-dimension, builds a concise sample of the database, such that with high probability, the execution of *any* query in the class on the sample provides an accurate estimate for the selectivity of the query on the original large database. The error probability holds *simultaneously* for the selectivity estimate of *all* queries in the collection, thus the same sample can be used to evaluate the selectivity of multiple queries, and the sample needs to be refreshed only following major changes in the database. The size of the sample does not depend on the size (number of tuples) in the database, just on the complexity of the class of queries we plan to run, measured by its VC-dimension. Both the analysis and the experimental results show that accurate selectivity estimates can be obtained using a sample of a surprisingly small size (see Table 7.2 for concrete values), which can then reside in main memory, with the net result of a significant speedup in the execution of queries on the sample.

A technical difficulty in applying the VC-dimension results to the database setting is that they assume the availability of a uniform sample of the Cartesian product of all the tables, while in practice it is more efficient to store a sample of each table separately and run the queries on the Cartesian product of the samples, which has a different distribution than a sample of the Cartesian product of the tables. We develop an efficient procedure for constructing a sample that circumvents this problem (see Sect. 7.4).

We present extensive experimental results that validate our theoretical analysis and demonstrate the advantage of our technique when compared to complex selectivity estimation techniques used in PostgreSQL and the Microsoft SQL Server. The main advantage of our method is that it gives

provably accurate predictions for the selectivities of all queries with up to a given complexity (VC-dimension) specified by the user before creating the sample, while techniques like multidimensional histograms or join synopses are accurate only for the queries for which they are built.

Note that we are only concerned with estimating the selectivity of a query, not with approximating the query answer using a sample of the database. Das [2009] presents a survey of the possible solutions to this latter task.

Outline. The rest of the chapter is organized as follows. We review the relevant previous work in Sect. 7.1. In Sect. 7.2 we give the necessary definition and formulate the problem of selectivity estimation. Our main analytical contribution, a bound on the VC dimension of class of queries is presented in Sect. 7.3. The application of these results for selectivity estimation is given in Sect. 7.4. Experiments are presented in Sect. 7.5.

7.1 Related work

Methods to estimate the selectivity (or cardinality of the output) of queries have been extensively studied in the database literature primarily due to the importance of this task to query plan optimization and resource allocation. A variety of approaches have been explored, ranging from the use of sampling, both online and offline, to the pre-computation of different statistics such as histograms, to the application of methods from machine learning [Chen et al., 1990; Harangsri et al., 1997], data mining [Gryz and Liang, 2004], optimization [Chaudhuri et al., 2007; Markl et al., 2007], and probabilistic modeling [Getoor et al., 2001; Ré and Suciu, 2010].

The use of sampling for selectivity estimation has been studied mainly in the context of online sampling [Lipton and Naughton, 1995; Lipton et al., 1990], where a sample is obtained, one tuple at a time, after the arrival of a query and it used only to evaluate the selectivity of that query and then discarded. Sampling at random from a large database residing on disk is an expensive operation [Brown and Haas, 2006; Gemulla et al., 2006; Olken, 1993], and in some cases sampling for an accurate cardinality estimate is not significantly faster than full execution of the query [Haas et al., 1993, 1994].

A variety of sampling and statistical analysis techniques has been tested to improve the efficiency of the sampling procedures and in particular to identify early stopping conditions. These include sequential sampling analysis [Haas and Swami, 1992; Hou et al., 1991], keeping additional statistics

to improve the estimation [Haas and Swami, 1995], labelling the tuples and using label-dependent estimation procedures [Ganguly et al., 1996], or applying the cumulative distribution function inversion procedure [Wu et al., 2001]. Some works also looked at non-uniform sampling [Babcock et al., 2003; Estan and Naughton, 2006] and stratified sampling [Chaudhuri et al., 2007; Joshi and Jermaine, 2008]. Despite all these relevant contributions, online sampling is still considered too expensive for most applications. An off-line sampling approach was explored by Ngu et al. [2004], who used systematic sampling (requiring the tuples in a table to be sorted according to one of the attributes) with a sample size dependent on the number of tuples in the table. Their work does not give any explicit guarantee on the accuracy of the predictions. Chaudhuri et al. [2007] present an approach which uses optimization techniques to identify suitable strata before sampling. The obtained sample is such that the mean square error in estimating the selectivity of queries belonging to a given workload is minimized, but there is no quality guarantee on the maximum error. Haas [1996] developed Hoeffding inequalities to bound the probability that the selectivity of a query estimated from a sample deviates more than a given amount from its expectation. However, to estimate the selectivity for multiple queries and obtain a given level accuracy for all of them, simultaneous statistical inference techniques like the union bound should be used, which are known to be overly conservative when the number of queries is large [Miller, 1981]. In contrast, our result holds simultaneously for *all* queries within a given complexity (VC-dimension).

A technical problem arises when combining join operations and sampling. As pointed out by Chaudhuri et al. [1999], the Cartesian product of uniform samples of a number of tables is different from a uniform sample of the Cartesian product of those tables. Furthermore, given a size s, it is impossible to a priori determine two sample sizes s_1 and s_2 such that uniform samples of these sizes from the two tables will give, when joined together along a common column, a sample of the join table of size s. In Sect. 7.4 we explain why only the first issue is of concern for us and how we circumvent it.

In practice most database systems use pre-computed statistics to predict query selectivity [Ganti et al., 2000; Gibbons and Matias, 1998; Hou et al., 1988; Jin et al., 2006; Larson et al., 2007], with histograms being the most commonly used representation. The construction, maintenance, and use of histograms were thoroughly examined in the literature [Ioannidis and Poosala, 1995; Jagadish et al., 1998; Matias et al., 1998; Poosala et al., 1996], with both theoretical and experimental results. In particular Chaudhuri et al. [1998] rigorously evaluated the size of the sample needed for

building a histogram providing good estimates for the selectivities of a large group of (select only, in their case) queries. Kaushik et al. [2005] extensively compared histograms and sampling from a space complexity point of view, although their sample-based estimator did not offer a uniform probabilistic guarantee over a set of queries and they only consider the case of foreign-key equijoins. We address both these points in our work. Although very efficient in terms of storage needs and query time, the quality of estimates through histograms is inherently limited for complex queries because of two major drawbacks in the use of histograms: intra-bucket uniformity assumption (i.e., assuming a uniform distribution for the frequencies of values in the same bucket) and inter-column independence assumption (i.e., assuming no correlation between the values in different columns of the same or of different tables). Different authors suggested solutions to improve the estimation of selectivity without making the above assumptions [Bruno and Chaudhuri, 2004; Dobra, 2005; Poosala and Ioannidis, 1997; Wang and Sevcik, 2003; Wang et al., 1997]. Among these solutions, the use of multidimensional histograms [Bruno et al., 2001; Poosala and Ioannidis, 1997; Srivastava et al., 2006; Wang and Sevcik, 2003] seems the most practical. Nevertheless, these techniques are not widespread due to the extra memory and computational costs in their implementation.

Efficient and practical techniques for drawing random samples from a database and for updating the sample when the underlying tables evolve have been extensively analyzed in the database literature [Brown and Haas, 2006; Gemulla et al., 2006, 2007; Haas and König, 2004; Jermaine et al., 2004].

In Sect. 2.1 we mentioned some uses of VC-dimension in the database literature. To the best of our knowledge, our work is the first to provide explicit bounds on the VC-dimension of SQL queries and to apply the results to query selectivity estimation.

7.2 Database queries and selectivity

We outline here some basic definitions about databases, queries, and selectivity. We refer the reader to complete textbooks for additional information [Garcia-Molina et al., 2002]. We consider a *database* $\mathcal{D}$ of k tables $\mathcal{T}_1, \ldots, \mathcal{T}_k$. A *table* is a two-dimensional representation of data. Its rows are called *tuples* and it can have multiple *columns*. We denote a column C of a table $\mathcal{T}$ as $\mathcal{T}.C$ and, for a tuple $t \in \mathcal{T}$, the value of t in the column C as $t.C$. We denote the domain of the values that can appear in a column $\mathcal{T}.C$ as $D(\mathcal{T}.C)$. Table 7.1 shows two examples of database tables, `Customers`

and `CarColors`. Our focus is on queries that combine select and join operations, defined as follows. We do not take projection operations into consideration because their selectivities have no impact on query optimization.

Table 7.1: Example of database tables.

Customers			
Name	*Street*	*ZipCode*	*PhoneNumber*
John Doe	Pratt Av.	02906	401-1234567
Jim Clark	Morris Rd.	02906	502-8902134
Greta Garbo	Pitman St.	05902	853-9876543

CarColor	
Name	*Color*
John Doe	Blue
Jane Doe	Red
Greta Garbo	Red

Definition 18. Given a table $\mathcal{T}$ with columns $\mathcal{T}.C_1, \ldots, \mathcal{T}.C_\ell$, a *selection query* q on $\mathcal{T}$ is an operation which returns a subset S of the tuples of $\mathcal{T}$ such that a tuple t of $\mathcal{T}$ belongs to S if and only if the values in t satisfy a condition $\mathcal{C}$ (the *selection predicate*) expressed by q. In full generality, $\mathcal{C}$ is the Boolean combination of clauses of the form $\mathcal{T}.C_i \,\mathrm{op}\, a_i$, where $\mathcal{T}.C_i$ is a column of $\mathcal{T}$, "op" is one of $\{<, >, \geq, \leq, =, \neq\}$ and a_i is an element of the domain of $\mathcal{T}.C_i$.

As an example, the selection query

$$\texttt{SELECT} * \texttt{FROM } \textit{Customers}, \texttt{WHERE } \textit{Customers.ZipCode} = 02906$$

would return the first and second row of Table 7.1.

We assume that all $D(\mathcal{T}.C_i)$ are such that it is possible to build total order relations on them. This assumptions does not exclude categorical domains from our discussion, because the only meaningful values for "op" for such domains are "=" and "$\neq$", so we can just assume an arbitrarily but fixed order for the categories in the domain.

Definition 19. Given two tables $\mathcal{T}_1$ and $\mathcal{T}_2$, a *join query* q on a common column C (i.e., a column present both in $\mathcal{T}_1$ and $\mathcal{T}_2$) is an operation which returns a subset of the Cartesian product of the tuples in $\mathcal{T}_1$ and $\mathcal{T}_2$. The returned subset is defined as the set

$$\{(t_1, t_2) \ : \ t_1 \in \mathcal{T}_1, t_2 \in \mathcal{T}_2, \text{ s.t. } t_1.C \,\mathrm{op}\, t_2.C\}$$

where "op" is one of $\{<, >, \geq, \leq, =, \neq\}$.

An example of a join query is the following:

$$\texttt{SELECT} * \texttt{FROM } \textit{Customers}, \textit{ CarColors } \texttt{WHERE } \textit{Customers.Name} = \textit{CarColors.Name}$$

This query would return the following tuples:

$$(\text{John Doe}, \text{Pratt Av.}, 02906, \text{401-1234567}, \text{Blue}),$$

$$(\text{Greta Garbo}, \text{Pitman St.}, 05902, \text{853-9876543}, \text{Red})$$

The column *Name* is reported only once for clarity.

Our definition of a join query is basically equivalent to that of a *theta-join* [Garcia-Molina et al., 2002, Sect.5.2.7], with the limitation that the join condition $\mathcal{C}$ can only contain a single clause, i.e., a single condition on the relationship of the values in the shared column C and only involve the operators $\{<, >, \geq, \leq, =, \neq\}$ (with their meaning on $D(C)$). The pairs of tuples composing the output of the join in our definition have a one-to-one correspondence with the tuples in the output of the corresponding theta-join.

Definition 20. Given a set of ℓ tables $\mathcal{T}_1, \ldots, \mathcal{T}_\ell$, a *combination of select and join operations* is a query returning a subset of the Cartesian product of the tuples in the sets $S_1, \ldots, S_\ell$, where S_i is the output of a selection query on $\mathcal{T}_i$. The returned set is defined by the selection queries and by a set of join queries on $S_1, \ldots, S_\ell$.

As an example, the query

$$\texttt{SELECT} * \texttt{FROM } \textit{Customers}, \textit{CarColors} \texttt{ WHERE } \textit{Customers.Name} = \textit{CarColors.Name}$$

$$\texttt{AND } \textit{Customers.ZipCode} = 02906 \texttt{ AND } \textit{CarColors.Color} = \textit{Red}$$

combines select and joins operations. It returns no tuple (empty answer), as there is no individual reported in both tables with zipcode 02906 and a red car.

Definition 21. Given a query q, a *query plan* for q is a directed binary tree T_q whose nodes are the elementary (i.e., select or join) operations into which q can be decomposed. There is an edge from a node a to a node b if the output of a is used as an input to b. The operations on the leaves of the tree use one or two tables of the database as input. The output of the operation in the root node of the tree is the output of the query.

It follows from the definition of a combination of select and join operations that a query may conform with multiple query plans. Nevertheless, for all the queries we defined there is (at least) one query plan such that all select operations are in the leaves and internal nodes are join nodes [Garcia-Molina et al., 2002]. To derive our results, we use these specific query plans.

Two crucial definitions that we use throughout the work are the *cardinality* of the output of a query and the equivalent concept of *selectivity* of a query.

Definition 22. Given a query q and a database $\mathcal{D}$, the *cardinality* of its output is the number of elements (tuples if q is a selection query, pairs of tuples if q is a join query, and ℓ-uples of tuples for combinations of join and select involving ℓ tables) in its output, when run on $\mathcal{D}$. The *selectivity* $\sigma(q)$ of q is the ratio between its cardinality and the product of the sizes of its input tables.

Our goal is to store a succinct representation (sample) $\mathcal{S}$ of the database $\mathcal{D}$ such that an execution of a query on the sample $\mathcal{S}$ will provide an accurate estimate for the selectivity of each operation in the query plan when executed on the database $\mathcal{D}$.

7.3 The VC-dimension of classes of queries

In this section we develop a general bound to the VC-dimension of classes of queries. We define appropriate range spaces $(D, \mathcal{R})$ as follows. For a class of select queries Q on a table $\mathcal{T}$, D is the set of all tuples in the input table, and $\mathcal{R}$ the family of the outputs (as sets of tuples) of the queries in Q when run on $\mathcal{T}$. For a class Q of queries combining select and join operations, D is the Cartesian product of the associated tables and $\mathcal{R}$ is the family of outcomes of queries in Q, seen as ℓ-uples of tuples, if ℓ tables are involved in the queries of Q. When the context is clear we identify the $\mathcal{R}$ with a class of queries.

When the ranges represent all the possible outputs of queries in a class Q applied to database tables $\mathcal{D}$, the VC-dimension of the range space is the maximum number of tuples such that any subset of them is the output of a query in Q.

In Sect. 7.4 we use an ε-approximation (Def. 2) to compute good estimates of the selectivities of all queries in Q. We obtain a small approximation set through probabilistic construction. The challenge in applying Thm. 1 to our setting is computing the VC-dimension of a range space defined by a class of queries.

We start by computing the VC-dimension of simple select queries on one column and then move to more complex queries (multi-attributes select queries, join queries). We then extend our bounds to general queries that are combinations of multiple select and join operations.

7.3.1 Select queries

Let $\mathcal{T}$ be a table with m columns $\mathcal{T}.C_1, \ldots, \mathcal{T}.C_m$, and n tuples. For a fixed column $\mathcal{T}.C$, consider the set Σ_C of the selection queries in the form

$$\texttt{SELECT} \ * \ \texttt{FROM} \ \mathcal{T} \ \texttt{WHERE} \ \mathcal{T}.C_i \operatorname{op} a \tag{7.1}$$

where op is an *inequality* operator (i.e., either "$\geq$" or "$\leq$")[1] and $a \in D(\mathcal{T}.C)$.

Let $q_1, q_2 \in \Sigma_C$ be two queries. We say that q_1 is equivalent to q_2 (and denote this fact as $q_1 = q_2$) if their outputs are identical, i.e., they return the same set of tuples when they are run on the same database. Note that $q_1 = q_2$ defines a proper equivalence relation.

Let $\Sigma^*_C \subseteq \Sigma_C$ be a maximum subset of Σ_C that contains no equivalent queries, i.e., it contains one query from each equivalent class.

Lemma 26. *Let $\mathcal{T}$ be a table with m columns C_i, $1 \leq i \leq m$, and consider the set of queries*

$$\Sigma^*_{\mathcal{T}} = \bigcup_{i=1}^{m} \Sigma^*_{C_i},$$

where $\Sigma^*_{C_i}$ is defined as in the previous paragraph. Then, the range space $S = (\mathcal{T}, \Sigma^*_{\mathcal{T}})$ has VC-dimension at most $m + 1$.

Proof. We can view the tuples of $\mathcal{T}$ as points in the m-dimensional space $\Delta = D(\mathcal{T}.C_1) \times D(\mathcal{T}.C_2) \times \ldots \times D(\mathcal{T}.C_m)$. A tuple $t \in \mathcal{T}$ such that $t.C_1 = a_1, t.C_2 = a_2, \ldots, t.C_m = a_m$ is represented on the space by the point $(a_1, a_2, \ldots, a_m)$.

The queries in $\Sigma^*_{\mathcal{T}}$ can be seen as half spaces of Δ. In particular any query in $\Sigma^*_{\mathcal{T}}$ is defined as in (7.1) and can be seen as the half space

$$\{(x_1, \ldots, x_i, \ldots, x_m) \ : \ x_j \in D(\mathcal{T}_j) \text{ for } j \neq i, \text{ and } x_i \operatorname{op} a_i\} \subseteq \Delta \ .$$

It then follows from Lemma 1 that $\mathsf{VC}(S) \leq m + 1$. □

We now extend these result to general selection queries. Consider the set $\Sigma^{2*}_{\mathcal{T}}$ of queries whose selection predicate can be expressed as the Boolean combination of the selection predicates of at most two queries from $\Sigma^*_{\mathcal{T}}$. These are the queries of the form:

$$\texttt{SELECT} \ * \ \texttt{FROM} \ \mathcal{T} \ \texttt{WHERE} \ \mathcal{T}.X_1 \operatorname{op}_1 a_1 \operatorname{bool} \mathcal{T}.X_2 \operatorname{op}_2 a_2$$

[1] The operators "$>$" and "$<$" can be reduced to "$\geq$" and "$\leq$" respectively.

where $\mathcal{T}.X_1$ and $\mathcal{T}.X_2$ are two columns from $\mathcal{T}$ (potentially, $\mathcal{T}.X_1 = \mathcal{T}.X_2$), $a_1 \in D(\mathcal{T}.X_1)$, $a_2 \in D(\mathcal{T}.X_2)$, "op$_i$" is either "$\geq$" or "$\leq$" and "bool" is either "`AND`" or "`OR`". Note that, in particular the queries in the form

$$\texttt{SELECT} \ * \ \texttt{FROM} \ \mathcal{T} \ \texttt{WHERE} \ \mathcal{T}.X_1 \, \text{eqop} \, a$$

where eqop is either "$=$" or "$\neq$", belong to $\Sigma^{2*}_{\mathcal{T}}$ because we can rewrite a selection predicate containing one of these operators as a selection predicate of two clauses using "$\geq$" and "$\leq$" joined by either AND (in the case of "$=$") or OR (in the case of "$\neq$").

By applying Lemma 2, we have that the VC-dimension of the range space $(\mathcal{T}, \Sigma^{2*}_{\mathcal{T}})$ is at most $3(m+1)2\log((m+1)2)$, where m is the number of columns in the table $\mathcal{T}$.

We can generalize this result to b Boolean combinations of selection predicates as follows.

Lemma 27. *Let $\mathcal{T}$ be a table with m columns, let $b > 0$ and let $\Sigma^{b*}_{\mathcal{T}}$ be the set of selection queries on $\mathcal{T}$ whose selection predicate is a Boolean combination of b clauses. Then, the VC-dimension of the range space $S_b = (\mathcal{T}, \Sigma^{b*}_{\mathcal{T}})$ is at most $3((m+1)b)\log((m+1)b)$.*

Note that we can not apply the bound used in the proof of Lemma 26: once we apply Boolean operations on the outputs of the individual select operation, the set of possible outputs, $S_b = (\mathcal{T}, \Sigma^{b*}_{\mathcal{T}})$, may form complex subsets, including unions of disjoint (half-open) axis aligned-rectangles and/or intersections of overlapping ones that cannot be represented as a collection of half spaces. Thus, we need to resort to a different technique.

Proof. The output of a query q in $\Sigma^{b^*}_{\mathcal{T}}$ can be seen as the Boolean combination (i.e., union and intersection) of the outputs of at most b "simple" select queries q_i from $\Sigma^{*}_{\mathcal{T}}$ where each of these queries q_i is as in (7.1). An `AND` operation in the predicate of q implies an intersection of the outputs of the corresponding two queries q_i and q_j, while an `OR` operation implies a union of the outputs. The thesis follows by applying Lemma 2. □

7.3.2 Join queries

Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two distinct tables, and let R_1 and R_2 be two families of (outputs of) select queries on the tuples of $\mathcal{T}_1$ and $\mathcal{T}_2$ respectively. Let $S_1 = (\mathcal{T}_1, R_1)$, $S_2 = (\mathcal{T}_2, R_2)$ and let $\mathsf{VC}(S_1), \mathsf{VC}(S_2) \geq 2$. Let C be a column along which $\mathcal{T}_1$ and $\mathcal{T}_2$ are joined, and let $T_J = \mathcal{T}_1 \times \mathcal{T}_2$ be the Cartesian product of the two tables.

For a pair of queries $r_1 \in R_1$, $r_2 \in R_2$, let

$$J^{\text{op}}_{r_1,r_2} = \{(t_1, t_2) \ : \ t_1 \in r_1, t_2 \in r_2, t_1.C \, \text{op} \, t_2.C\},$$

where op $\in \{>, <, \geq, \leq, =, \neq\}$. $J^{\text{op}}_{r_1,r_2}$ is the set of ordered pairs of tuples (one from $\mathcal{T}_1$ and one from $\mathcal{T}_2$ that forms the output of the join query

$$\texttt{SELECT} \ * \ \texttt{FROM} \ \mathcal{T}_1, \mathcal{T}_2 \ \texttt{WHERE} \ r_1 \ \texttt{AND} \ r_2 \ \texttt{AND} \ \mathcal{T}_1.C \operatorname{op} \mathcal{T}_2.C \ . \tag{7.2}$$

Here we simplify the notation by identifying select queries with their predicates. We have $J^{\text{op}}_{r_1,r_2} \subseteq r_1 \times r_2$ and $J^{\text{op}}_{r_1,r_2} \subseteq T_J$. Let

$$J_C = \{J^{\text{op}}_{r_1,r_2} \mid r_1 \in R_1, r_2 \in R_2, \text{op} \in \{>, <, \geq, \leq, =, \neq\}\}.$$

J_C is the set of outputs of all join queries like the one in (7.2), for all pairs of queries in $R_1 \times R_2$ and all values of "op". We present here an upper bound to the VC-dimension of the range space $S_J = (T_J, J_C)$.

Lemma 28. $\mathsf{VC}(S_J) \leq 3(\mathsf{VC}(S_1) + \mathsf{VC}(S_2)) \log((\mathsf{VC}(S_1) + \mathsf{VC}(S_2)))$.

Proof. Let $v_1 = \mathsf{VC}(S_1)$ and $v_2 = \mathsf{VC}(S_2)$. Assume that a set $A \subseteq T_J$ is shattered by J_C, and $|A| = v$. Consider the two cross-sections $A_1 = \{x \in \mathcal{T}_1 \ : \ (x, y) \in A\}$ and $A_2 = \{y \in \mathcal{T}_2 \ : \ (x, y) \in A\}$. Note that $|A_1| \leq v$ and $|A_2| \leq v$ and by [Alon and Spencer, 2008, Corol. 14.4.3], $|P_{R_1}(A_1)| \leq g(v_1, v) \leq v^{v_1}$ and $|P_{R_2}(A_2)| \leq g(v_2, v) \leq v^{v_2}$. For each set $r \in P_{J_C}(A)$ (i.e., for each subset $r \subseteq A$, given that $P_{J_C}(A) = 2^A$) there is a pair (r_1, r_2), $r_1 \in R_1$, $r_2 \in R_2$, and there is op, such that $r = A \cap J^{\text{op}}_{r_1,r_2}$. Each of such pair (r_1, r_2) identifies a distinct pair $(r_1 \cap A_1, r_2 \cap A_2) \in P_{R_1}(A_1) \times P_{R_2}(A_2)$, therefore each element of $P_{R_1}(A_1) \times P_{R_2}(A_2)$ can be identified at most 6 times, the number of possible values for "op".

In particular, for a fixed "op", an element of $P_{R_1}(A_1) \times P_{R_2}(A_2)$ can be identified at most once. To see this, consider two different sets $s_1, s_2 \in P_{J_C}(A)$. Let the pairs (a_1, a_2), (b_1, b_2), $a_1, b_1 \in R_1$, $a_2, b_2 \in R_2$, be such that $s_1 = A \cap J^{\text{op}}_{a_1,a_2}$ and $s_2 = A \cap J^{\text{op}}_{b_1,b_2}$. Suppose that $a_1 \cap A_1 = b_1 \cap A_1$ $(\in P_{R_1}(A_1))$ and $a_2 \cap A_2 = b_2 \cap A_2$ $(\in P_{R_2}(A_2))$. The set s_1 can be seen as $\{(t_1, t_2) \ : \ t_1 \in a_1 \cap A_1, t_2 \in a_2 \cap A_2 \text{ s.t. } t_1.C \operatorname{op} t_2.C\}$. Analogously the set s_2 can be seen as $\{(t_1, t_2) \ : \ t_1 \in b_1 \cap A_1, t_2 \in b_2 \cap A_2 \text{ s.t. } t_1.C \operatorname{op} t_2.C\}$. But given that $a_1 \cap A_1 = b_1 \cap A_1$ and $a_2 \cap A_2 = b_2 \cap A_2$, this leads to $s_1 = s_2$, a contradiction. Hence, a pair (c_1, c_2), $c_1 \in P_{R_1}(A_1)$, $c_2 \in P_{R_2}(A_2)$ can only be identified at most 6 times, one for each possible value of "op".

Thus,$|P_{J_C}(A)| \leq 6|P_{R_1}(A_1)| \cdot |P_{R_2}(A_2)|$. A could not be shattered if $|P_{J_C}(A)| < 2^v$. Since

$$|P_{J_C}(A)| \leq 6|P_{R_1}(A_1)| \cdot |P_{R_2}(A_2)| \leq 6g(v_1, v)g(v_2, v) \leq 6v^{v_1+v_2},$$

it is sufficient to have $6v^{v_1+v_2} \leq 2^v$, which holds for $v > 3(v_1 + v_2)\log(v_1 + v_2)$. □

The relative complexity of this proof is due to the fact that the result of a join between the

output of two selection queries obviously depends on the two selection predicates, and we can not ignore this.

The above results can be generalized to any query plan represented as a tree where the select operations are in the leaves and all internal nodes are join operations. As we said earlier, such a tree exists for any query.

Lemma 29. *Consider the class Q of queries that can be seen as combinations of select and joins on $u > 2$ tables $\mathcal{T}_1, \ldots, \mathcal{T}_u$. Let $S_i = (\mathcal{T}_i, R_i)$, $i = 1, \ldots, u$ be the range space associated with the select queries on the u tables. Let $v_i = \mathsf{VC}(S_i)$. Let m be the maximum number of columns in a table $\mathcal{T}_i$. We assume $m \leq \sum_i v_i$.*[2] *Let $S_Q = (\mathcal{T}_1 \times \cdots \times \mathcal{T}_u, R_Q)$ be the range space associated with the class Q. The range set R_Q is defined as follows. Let $\rho = (r_1, \ldots, r_u)$, $r_i \in R_i$, and let ω be a sequence of $u-1$ join conditions representing a possible way to join the u tables $\mathcal{T}_i$, using the operators $\{>, <, \geq, \leq, =, \neq\}$. We define the range*

$$J_\rho^\omega = \{(t_1, \ldots, t_u) \ : \ t_i \in r_i, \ \textit{s.t.} \ (t_1, \ldots, t_u) \ \textit{satisfies} \ \omega\}.$$

R_Q is the set of all possible J_ρ^ω. Then,

$$\mathsf{VC}(S_Q) \leq 4u(\sum_i \mathsf{VC}(S_i)) \log(u \sum_i \mathsf{VC}(S_i)).$$

Note that this Lemma is not just an extension of Lemma 28 to join queries between two multicolumns tables. Instead, it is an extension to queries containing joins between multiple tables (possibly between multicolumns tables).

Proof. Assume that a set $A \subseteq T_J$ is shattered by R_Q, and $|A| = v$. Consider the cross-sections $A_i = \{x \in \mathcal{T}_i \ : \ (y_1, \ldots, y_{i-1}, x, y_{i+1}, \ldots, y_u) \in A\}, 1 \leq i \leq u$. Note that $|A_i| \leq v$ and by [Alon and Spencer, 2008, Coroll. 14.4.3] $|P_{R_i}(A_i)| \leq g(v_i, v) \leq v^{v_i}$. For each set $r \in P_{J_C}(A)$ (i.e., for each subset $r \subseteq A$, given that $P_{J_C}(A) = 2^A$) there is a sequence $\rho = (r_1, \ldots, r_u)$, $r_i \in R_i$, and there is an ω, such that $r = A \cap J_\rho^\omega$. Each sequence ρ identifies a distinct sequence $(r_1 \cap A_1, r_2 \cap A_2, \ldots, r_u \cap A_u) \in P_{R_1}(A_1) \times \cdots \times P_{R_u}(A_u)$, therefore each element of $P_{R_1}(A_1) \times \cdots \times P_{R_u}(A_u)$ can be identified at most 6^{u-1} times, one for each different ω.

In particular, for a fixed ω, an element of $P_{R_1}(A_1) \times \cdots \times P_{R_u}(A_u)$ can be identified at most once. To see this, consider two different sets $s_1, s_2 \in P_{J_C}(A)$. Let the vectors $\rho_a = (a_1, \ldots, a_u)$, $\rho_b = (b_1, \ldots, b_u)$, $a_i, b_i \in R_i$, be such that $s_1 = A \cap J_{\rho_a}^\omega$ and $s_2 = A \cap J_{\rho_b}^\omega$. Suppose that $a_i \cap A_i = b_i \cap A_i$ $(\in P_{R_i}(A_i))$. The set s_1 can be seen as $\{(t_1, \ldots, t_u) \ : \ t_i \in a_i \cap A_i, \text{ s.t. } (t_1, \ldots, t_u) \text{ satisfies } \omega\}$. Analogously the set s_2 can be seen as $\{(t_1, \ldots, t_u) \ : \ t_i \in b_i \cap A_i, \text{ s.t. } (t_1, \ldots, t_u) \text{ satisfies } \omega\}$. But given that $a_i \cap A_i = b_i \cap A_i$, this leads to $s_1 = s_2$, a contradiction. Hence, a vector $(c_1, \ldots, c_u)$, $c_i \in P_{R_i}(A_i)$, can only be identified at most a finite number ℓ of times, once for each different ω.

[2] The assumption $m \leq \sum_i v_i$ is reasonable for any practical case.

For each of the $u-1$ join conditions composing ω we need to choose a pair $(\mathcal{T}_1.A, \mathcal{T}_1.B)$ expressing the columns along which the tuples should be joined. There are at most $g = \binom{um}{2}$ such pairs (some of them cannot actually be chosen, e.g., those of the type $(\mathcal{T}_1.A, \mathcal{T}_1.B)$). There are then $\binom{g}{u-1}$ ways of choosing these $u-1$ pairs. For each choice of $u-1$ pairs, there are $6^{(u-1)}$ ways of choosing the operators in the join conditions (6 choices for op for each pair). We have

$$\ell \leq \binom{\binom{um}{2}}{u-1} \cdot 6^{(u-1)} \leq (mu)^{2u}.$$

Thus, $|P_{J_C}(A)| \leq \ell |P_{R_1}(A_1)| \cdot \cdots \cdot |P_{R_u}(A_u)|$. A could not be shattered if $|P_{J_C}(A)| < 2^v$. Since we have

$$|P_{J_C}(A)| \leq \ell \cdot |P_{R_1}(A_1)| \cdot \cdots \cdot |P_{R_u}(A_u)| \leq \ell \cdot g(v_1, v)g(v_2, v) \ldots g(v_u, v) \leq$$
$$\leq (mu)^{2u} \cdot v^{v_1 + \cdots + v_u},$$

then it is sufficient to have

$$(mu)^{2u} v^{\sum_{i=1}^{v} vi} \leq 2^v,$$

which holds for $v > 4u\left(\sum_i v_i\right) \log(u \sum_i v_i)$. □

7.3.3 Generic queries

Combining the above results we prove:

Theorem 9. *Consider a class $Q_{u,m,b}$ of all queries with up to $u-1$ join and u select operations, where each select operation involves no more than m columns and b Boolean operations, then*

$$\mathsf{VC}(Q_{u,m,b}) \leq 12u^2(m+1)b \log((m+1)b) \log(3u^2(m+1)b \log((m+1)b)).$$

Note that Thm. 9 gives an upper bound to the VC-dimension. Our experiments suggest that in most cases the VC-dimension and the corresponding minimum sample size are even smaller.

7.4 Estimating query selectivity

We apply the theoretical result on the VC-dimension of queries to constructing a concrete algorithm for selectivity estimation and query plan optimization.

7.4.1 The general scheme

Our goal is to apply Def. 2 and Thm. 1 to compute an estimate of the selectivity of SQL queries. Let $Q_{u,m,b}$ be a class of queries as in Thm. 9. The class $Q_{u,m,b}$ defines a range space $S = (D, \mathcal{R})$

such that D is the Cartesian product of the tables involved in executing queries in $Q_{u,m,b}$, and $\mathcal{R}$ is the family of all output sets of queries in $Q_{u,m,b}$. Let $\mathcal{S}$ be an ε-approximation of S and let R be the output set of a query $q \in Q_{u,m,b}$ when executed on the original dataset, then $D \cap R = R$ and $R \cap \mathcal{S}$ is the output set when the query is executed on the sample (see details below). Thus, by Def. 2,

$$\left| \frac{|D \cap R|}{|D|} - \frac{|\mathcal{S} \cap R|}{|\mathcal{S}|} \right| = |\sigma_{\mathcal{D}}(q) - \sigma_{\mathcal{S}}(q)| \leq \varepsilon,$$

i.e., the selectivity of a query $q \in Q_{u,m,b}$ on an ε-approximation of $(D, \mathcal{R})$ is within ε of the selectivity of q when executed on the original set of tables. Note that for any execution plan of a query $q \in Q_{u,m,b}$, all the queries that correspond to subtrees rooted at internal nodes of the plan are queries in $Q_{u,m,b}$. Thus, by running query q on an ε-approximation of $(D, \mathcal{R})$ we obtain accurate estimates for the selectivity of all the subqueries defined by its execution plan.

7.4.2 Building and using the sample representation

We apply Thm. 1 to probabilistically construct an ε-approximation of $(D, \mathcal{R})$. A technical difficulty in algorithmic application of the theorem is that it is proven for a uniform sample of the Cartesian product of all the tables in the database, while in practice it is more efficient to maintain the table structure of the original database in the sample. It is easier to sample each table independently, and to run the query on a sample that consists of subsets of the original tables rather than re-writing the query to run on a Cartesian product of tuples. However, the Cartesian product of independent uniform samples of tables is not a uniform sample of the Cartesian product of the tables [Chaudhuri et al., 1999]. We developed the following procedure to circumvent this problem. Assume that we need a uniform sample of size t from $\mathbf{D}$, which is the Cartesian product of ℓ tables $\mathcal{T}_1, \ldots, \mathcal{T}_\ell$. We then sample t tuples uniformly at random (with replacement) from each table $\mathcal{T}_i$, to form a sample table $\mathcal{S}_i$. We add an attribute *sampleindex* to each $\mathcal{S}_i$ and we set the value in the added attribute for each tuple in $\mathcal{S}_i$ to a unique value in $[1, t]$. Now, each sample table will contain t tuples, each tuple with a different index value in $[1, t]$. Given an index value $i \in [1, t]$, consider the set of tuples $X_i = \{x_1, \ldots, x_\ell\}$, $x_j \in \mathcal{S}_i$ such that $x_1.sampleindex = x_2.sampleindex = \cdots = x_\ell.sampleindex = i$. X_i can be seen as a tuple sampled from $\mathbf{D}$, and the set of all X_i, $i \in [1, t]$ is a uniform random sample of size t from $\mathbf{D}$. We run queries on the sample tables, but in order to estimate the selectivity of a join operation we count a tuple Y in the result only if the set of tuples composing Y is a subset of X_i for some $i \in [1, t]$. This is easily done by scanning the results and checking the values in the

sampleindex columns (see Algorithms 5 and 6).

Algorithm 5: `CreateSample`$(s, (T_1, \ldots, T_k))$

input : sample size s, tables $\mathcal{T}_1, ..., \mathcal{T}_k$.
output: sample tables $\mathcal{S}_1, ..., \mathcal{S}_k$ with t tuples each.

```
for j ← 1 to k do
    S_j ← ∅
end
for i ← 1 to s do
    for j ← 1 to k do
        t ← drawRandomTuple(T_j)
        t.sampleindex_j ← i
        S_j ← S_j ∪ {t}
    end
end
```

Algorithm 6: `ComputeSelectivity`$(\mathcal{S}, op)$

input : elementar database operation op, sample database $S = (\mathcal{S}_1, \ldots, \mathcal{S}_k)$ of size s.
output: the selectivity of op.

```
O_op ← executeOperation(S, op);
(ℓ_1, ..., ℓ_j) ← indexes of the sample tables involved in op ;
i ← 0;
for tuple ∈ O_op do
    if tuple.sampleindex_ℓ1 = tuple.sampleindex_ℓ2 = ··· = tuple.sampleindex_ℓj then
    i ← i + 1;
end
selectivity ← i/s;
```

Theorem 10. *The* `ComputeSelectivity` *procedure (in Alg. 6) executes a query on the Cartesian product of independent random samples of the tables but outputs the selectivity that corresponds to executing the query on a random sample of the Cartesian product of the original tables.*

Proof. The *CreateSample* procedure chooses from each table a random sample of t tuples and adds to each sampled tuple an index in $[1, t]$. Each sample table has exactly one tuple with each index value, and the Cartesian product of the sample tables has exactly one element that is a concatenation of tuples, all with the same index i in their tables. Restricting the selectivity computation to these t elements (as in *ComputeSelectivity*) gives the result. □

Note that our method circumvent the major difficulty pointed out by Chaudhuri et al. [1999]. They also proved that, in general, it is impossible to predict sample sizes for given two tables such that the join of the samples of two tables will result in a sample of a required size out of the join of the two tables. Our method does not require a sample of a given size from the result of a join. The

VC-dimension sampling technique requires only a sample of a given size from the Cartesian product of the tables, which is guaranteed by the above procedure.

Identifying the optimal query plan during query optimization may require executing several candidate query plans on the sample. A standard bottom-up candidate plan generation allows us to execute sub-plans once, store their results and reuse them multiple times as they will be common to many candidate plans. While the overhead of this execution-based selectivity estimation approach will still likely be higher than that of pre-computation based techniques (e.g., histograms), the reduced execution times of highly optimized plans enabled by better estimates, especially for complex and long-running queries, will more than compensate for this overhead. Thus, storing intermediate results that are common to several executions will speed up the total execution time on the sample. The significant improvement in the selectivity estimates in complex queries well compensates for the extra work in computing the selectivity estimates.

7.5 Experimental evaluation

This section presents the results of the experiments we run to validate our theoretical results and to compare our selectivity estimation method with standard techniques implemented in PostgreSQL and in Microsoft SQL Server.

Goals. The first goal of the experiments is to evaluate the practical usefulness of our theoretical results. To assess this, we run queries on a large database and on random samples of different sizes. We use the selectivity of the each query in the random samples as an estimator for the selectivity in the large database with the adjustments for join operations, as described in the previous Section. We compute the error between the estimate and the actual selectivity to show that the thesis of Thm. 1 is indeed valid in practice. The use of a large number of queries and of a variety of parameters allows us to evaluate the error rate as a function of the sample size. We then compare our method with the commonly used selectivity estimation based on precomputed histograms (briefly described in Sect. 7.5.1). We use histograms with a different number of buckets to show that, no matter how fine-grained the histograms might be, as soon as the inter-column and intra-bucket assumptions are no longer satisfied, our approach gives better selectivity estimates.

7.5.1 Selectivity estimation with histograms

In many modern database systems, the query optimizer relies on histograms for computing data distribution statistics to help determine the most efficient query plans. In particular, PostgreSQL uses one-dimensional equi-depth (i.e., equal frequency buckets) histograms and a list of the most common values (MCV) for each column (of a database table) to compute optimizer statistics. The MCV information stores the most frequent N items (by default $N = 100$) and their frequency for each column. The histograms (by default with 100 bins) are built for the values not stored in the MCV list. The selectivity of a constraint $A = x$, where A is a column and x is a value is computed from the MCV list if x is in the MCV list or from the histogram bin that contains x if x is not in the MCV list. The selectivity of a range constraint such as $A < x$ is computed with information from both the MCV list and the histogram, i.e., the frequencies of the most common values less than x and the frequency estimate for $A < x$ from the histogram will be added to obtain the selectivity.

In PostgreSQL, the histograms and the MCV lists for the columns of a table are built using a random sample of the tuples of the table. The histograms and the MCV list for all columns of a table are based on the same sample tuples (and are therefore correlated). The sample size is computed for each column using a formula based on the table size, histogram size, and a target error probability developed by Chaudhuri et al. [1998] and the largest sample size required by the columns of a table is used to set the sample size of the table.

Finally, the join selectivity of multiple constraints are computed using the attribute independence assumption: e.g., selectivities are added in case of an OR operator and multiplied for an AND operator. Therefore, large selectivity estimation errors are possible for complex queries and correlated inputs.

7.5.2 Setup

Original tables. The tables in our large database were randomly generated and contain 20 million tuples each. There is a distinction between tables used for running selection queries and tables used for running join (and selection) queries. For tables on which we run selection queries only, the distributions of values in the columns fall in two different categories:

- **Uniform and Independent:** The values in the columns are chosen uniformly and independently at random from a fixed domain (the integer interval $[0, 200000]$, the same for all

columns). Each column is treated independently from the others.

- **Correlated:** Two columns of the tables contain values following a multivariate normal distribution with mean $M = \mu\mathbb{I}_{2,2}$ and a non-identity covariance matrix Σ (i.e., the values in the two different columns are correlated).

The tables for join queries should be considered in pairs (A, B) (i.e., the join happens along a common column C of tables A and B). The values in the columns are chosen uniformly and independently at random from a fixed domain (the integer interval $[0, 200000]$, the same for all columns). Each column is treated independently from the others.

Sample tables. We sampled tuples from the large tables uniformly, independently, and with replacement, to build the sample tables. For the samples of the tables used to run join queries, we drew random tuples uniformly at random from the base tables independently and added a column *sampleindex* to each tuple such that each tuple drawn from the same base table has a different value in the additional column and with tuples from different tables forming an element of the sample (of the Cartesian product of the base tables) if they have the same value in this additional column, as described in Sect. 7.4.2.

For each table in the original database we create many sample tables of different sizes. The sizes are either fixed arbitrarily or computed using (2.2) from Thm. 1. The arbitrarily sized sample tables contain between 10000 and 1.5 million tuples. To compute the VC-dimension-dependent sample size, we fixed $\varepsilon = 0.05$, $\delta = 0.05$, and $c = 0.5$. The parameter d was set to the best bound to the VC-dimension of the range space of the queries we were running, as obtained from our theoretical results. If we let m be the number of columns involved in the selection predicate of the queries and b be the number of Boolean clauses in the predicate, we have that d depends directly on m and b, as does the sample size s through (2.2) in Thm. 1. For selection queries, we used $m = 1, 2$ and $b = 1, 2, 3, 5, 8$, with the addition of the combination $m = 5$, $b = 5$. We run experiments on join queries only for some combinations of m and b (i.e., for $m = 1$ and $b = 1, 2, 5, 8$) due to the large size of the resulting sample tables. Table 7.2 shows the sample sizes, as number of tuples, for the combinations of parameters we used in our experiments.

Table 7.2: VC-Dimension bounds and samples sizes, as number of tuples, for the combinations of parameters we used in our experiments.

		Query type			
		Select		Join	
Columns (m)	Boolean clauses (b)	VC-dim	Sample size	VC-dim	Sample size
1	1	2	1000	4	1400
	2	4	1400	16	3800
	3	6	2800	36	7800
	5	10	2600	100	20600
	8	16	3800	256	51800
2	2	31	6800		
	3	57	12000		
	5	117	24000		
	8	220	44600		
5	5	294	59400		

Histograms. We built histograms with a different number of buckets, ranging from 100 to 10000. Due to limitations in PostgreSQL, incrementing the number of buckets in the histograms also increments the number of values stored in the MCV list. Even if this fact should have a positive influence on the quality of the selectivity estimates obtained from the histograms, our results show that the impact is minimal, especially when the inter-column independence and the intra-bucket uniformity assumptions are not satisfied. For SQL Server, we built the standard single-column histograms and computed the multi-column statistics which should help obtaining better estimations when the values along the columns are correlated.

Queries. For each combination of the parameters m and b and each large table (or pair of large tables, in the case of join) we created 100 queries, with selection predicates involving m columns and b Boolean clauses. The parameters in each clause, the range quantifiers, and the Boolean operators connecting the different clauses were chosen uniformly at random to ensure a wide coverage of possible queries.

7.5.3 Results

Selection queries. The first result of our experiments is that, for all the queries we ran, on all the sample tables, the estimate of the selectivity computed using our method was within ε ($= 0.05$) from the real selectivity. The same was not true for the selectivity computed by the histograms. As

an example, in the case of $m = 2$, $b = 5$ and uniform independent values in the columns, the default PostgreSQL histograms predicted a selectivity more than ε off from the real selectivity for 30 out of 100 queries. Nevertheless, in some of cases the histograms predicted a selectivity closer to the actual one than what our method predicted. This is especially true when the histogram independence assumption holds (e.g., for $m = 2$, $b = 5$ the default histograms gave a better prediction than our technique in 11 out of 100 cases). Similar situations also arise for SQLServer.

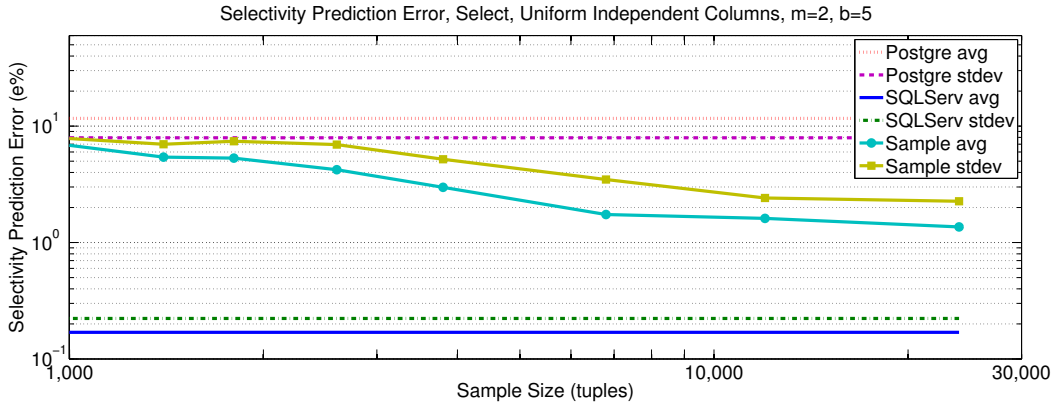


Figure 7.1: Selectivity prediction error for selection queries on a table with uniform independent columns – Two columns ($m = 2$), five Boolean clauses ($b = 5$).

Since the selectivity estimated by the our method was always within ε from the actual, we report the actual percent error, i.e., the quantity $e_{\%} = \frac{100|p(\sigma_q) - \sigma_{\mathcal{D}}(q)|}{\sigma_{\mathcal{D}}(q)}$ where $p(\sigma_q)$ is the predicted selectivity. We analyze the average and the standard deviation of this quantity on a set of queries and the evolution of these measures as the sample size increases. We can see from Fig. 7.1 and 7.2 that both the average and the standard deviation of the percentage error of the prediction obtained with our method decrease as the sample size grows (the rightmost plotted sample size is the one from Table 7.2, i.e., the one computed in Thm.1. More interesting is the comparison in those figures between the performance of the histograms and the performance of our techniques in predicting selectivities. When the assumptions of the histograms hold, as is the case for the data plotted in Fig. 7.1, the predictions obtained from the histograms are good.

But as soon as the data are correlated (Fig. 7.2), our sampling method gives better predictions than the histograms even at the smallest sample sizes and keeps improving as the sample grows larger. It is also interesting to observe how the standard deviation of the prediction error is much smaller for our method than for the histograms, suggesting a much higher consistency in the quality

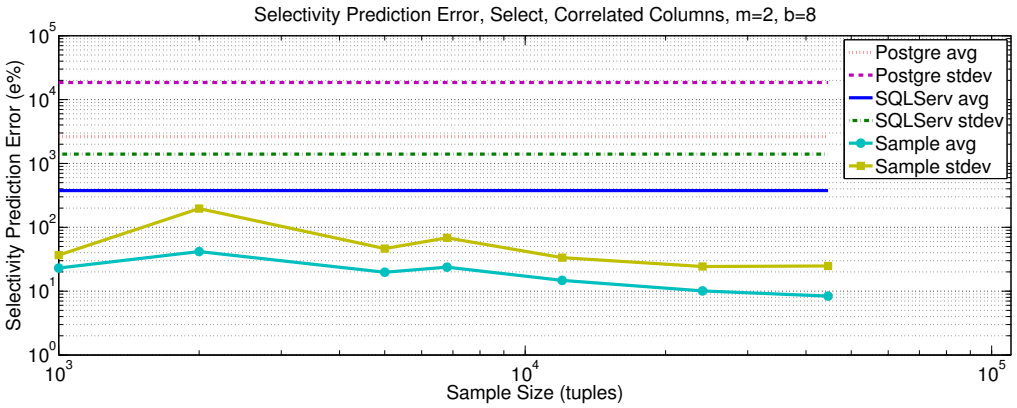


Figure 7.2: Selectivity prediction error for selection queries on a table with correlated columns – Two columns ($m = 2$), eight Boolean clauses ($b = 8$).

of the predictions. In Fig. 7.2 we do not show multiple curves for the different PostgreSQL histograms because increasing the number of buckets had very marginal impact on the quality of the estimates, sometimes even in the negative sense (i.e., an histogram with more buckets gave worse predictions than an histogram with fewer buckets), a fact that can be explained with the variance introduced by the sampling process used to create the histograms. For the same reason we do not plot multiple lines for the prediction obtained from the multi-columns and single-column statistics of SQL Server: even when the multi-column statistics were supposed to help, as in the case of correlated data, the obtained prediction were not much different from the ones obtained from the single-column histograms.

Join queries. The strength of our method compared to histograms is even more evident when we run join queries, even when the histograms independent assumptions are satisfied. In our experiments, the predictions obtained using our technique were always within ε from the real values, even at the smallest sample sizes, but the same was not true for histograms. For example, in the case of $m = 1$ and $b = 5$, 135 out of 300 predictions from the histograms were more than ε off from the real selectivities. Figure 7.3 shows the comparison between the average and the standard deviation of the percentage error, defined in the previous paragraph, for the histograms and our method. The numbers include predictions for the selection operations at the leaves of the query tree.

Again, we did not plot multiple curves for histograms with a different number of buckets because the quality of the predictions did not improve as the histograms became more fine-grained. To understand the big discrepancy between the accurate predictions of our method and the wrong

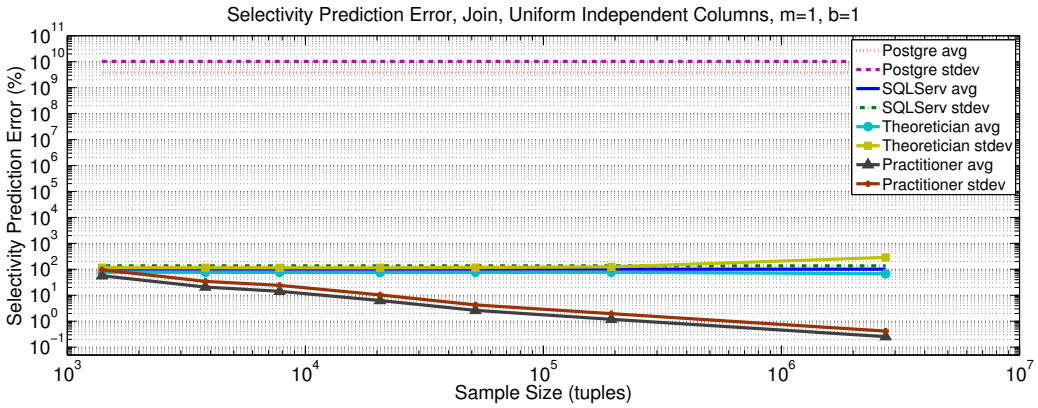


Figure 7.3: Selectivity prediction error for join queries queries. One column ($m = 1$), one Boolean clause ($b = 1$).

estimates computed by the histograms in PostgreSQL we note that for some join queries, the histograms predicted an output size on the order of the hundreds of thousands of tuples but the actual output size was zero or a very small number of tuples. Observing the curves of the average and the standard deviation of the percentage error for the prediction obtained with our method, we can see that at the smaller sample sizes the quality of the predictions only improves minimally with the sample size. This is due to the fact that at small sizes our prediction for the join operation is very often zero or very close to zero, because the output of the query does not contain enough pairs of tuples from the sample of the Cartesian product of the input table (i.e., pairs of tuples with the same value in the *sampleindex* column). In these cases, the prediction can not be accurate at all (i.e., the error is 100% if the original output contained some tuples, or 0% if the query returned an empty set in the large databases). As soon as the sample size grows more, we can see first a jump to higher values of the percentage error, which then behaves as expected, i.e., decreasing as the sample size increases.

In Fig. 7.3 we also show a comparison between the percentage error of predictions obtained using our method in two different ways: the "theoretically correct" way that makes use of the number of pairs of tuples with the same value in the *sampleindex* column and the "practitioner" way which uses the size of the output of the join operation in the sample, therefore ignoring the *sampleindex* column. Recall that we had to add the *sampleindex* column because Thm. 1 requires a uniform sample of the Cartesian product of the input tables. As it is evident from Fig. 7.3, the "practitioner" way of predicting selectivity gives very good results at small sample sizes (although

it does not offer theoretical guarantees). These results are similar in spirit, although not equivalent, to the theoretical conclusions presented by Haas et al. [1996] in the setting of selectivity estimation using online sampling.

7.6 Conclusions

We develop a novel method for estimating the selectivity of queries by executing it on a concise, properly selected, sample of the database. We present a rigorous analysis of our method and extensive experimental results demonstrating its efficiency and the accuracy of its predictions.

Most commercial databases use histograms built on a single column, for selectivity estimation. There has also been significant research on improving the estimate using multidimensional histograms [Bruno et al., 2001; Poosala and Ioannidis, 1997; Srivastava et al., 2006; Wang and Sevcik, 2003] and join synopses [Acharya et al., 1999]. The main advantage of our method is that it gives uniformly accurate estimates for the selectivity of any query within a predefined VC-dimension range. Methods that collect and store pre-computed statistics give accurate estimates only for the relations captured by the collected statistics, while estimate of any other relation relies on an independence assumption. To match the accuracy of our new method with histograms and join synopses one would need to create, for each table, a multidimensional histogram where the number of dimensions is equal to the number of columns in the tables. The space needed for a multidimensional histogram is exponential in the number of dimensions, while the size of our sample representation is almost linear in that parameter. Furthermore, to estimate the selectivity for join operations one would need to create join synopses for all pairs of columns in the database, again in space that grows exponential in the number of columns.

VC-dimension, often considered only a theoretical tool, leads to an efficient and practical tool for an important problem in database management.

Chapter 8

Conclusions and Future Work

The major goal of the work that led to this dissertation was to verify whether VC-dimension could be used to develop practical sample size bounds for important data analytics problems. We showed that it is possible to compute tight upper bounds to the VC-dimension of relevant problems from different areas of data analytics: knowledge discovery, graph analysis, and database management.

The upper bounds that we found are characteristic quantities of the dataset or of the problem at hands. Not only they can be easy to compute, but they are usually small in practice. This means that the sample size needed to obtain high quality approximations of the results for the problem at hand may not need to depend on the size of the dataset or on the number of patterns or "questions" that one is interested to answer using the data. The sample sizes obtained using our bounds are small and can fit into main memory, partially solving the issue of analyzing very large datasets. Indeed, the practicality of our results is exemplified by the fact that one of our algorithms (PARMA, see Ch. 4) was adapted and implemented to be used in SAMOA [De Francisci Morales, 2013] at Yahoo (see also `http://yahoo.github.io/samoa/`).

Despite the many advantages of using VC-dimension when deriving sample sizes for randomized algorithms, it should not be considered a "one-size-fits-all method". Indeed, its applicability at large may be hindered by the following factors:

- **Need for an efficient algorithm to compute bounds to the VC-dimension.** Bounding the VC-dimension of a problem is challenging but good bounds can be found. Nevertheless, even if the bound is tight and it is a characteristic quantity of the dataset or of the problem at hand, actually computing this quantity may be computational expensive. We saw an example

of this issue in Sect. 6.4.1, when dealing with the computation of the vertex-diameter of a directed or weighted graph. In that case, we could use a fast-to-compute loose approximation of the vertex diameter thanks to the presence of the logarithm of the vertex diameter in the bound of the VC-dimension from Lemma 15. Without a fast procedure to compute the upper bound to the VC-dimension (and therefore the needed sample size), the use of VC-dimension is severely limited.

- **Need for an efficient sampling procedure.** A bound to the VC-dimension is not sufficient: the second necessary component is a sampling procedure to sample points of the domain according to the appropriate probability distribution. We described an example of how to develop such a sampling procedure in Sect. 6.4.1. It is of crucial importance that the sampling procedure is fast, otherwise many advantages of using sampling may be lost.

- **Need of drawing independent samples.** In order to apply Thm. 1, we need *independent* samples from the distribution. At times, this may be a limitation. For example we saw in Sect. 6.4.3 that the algorithm from [Brandes and Pich, 2007] draws multiple shortest paths at each step but they are dependent, as they all originate from the same vertex. By doing this, the algorithm collects more information, resulting in low variance in the estimation, but we were not able to show that it actually computes an ε-approximation to the betweenness of all the vertices. Recent developments extends Thm. 1 to the case of ergodic sampling, therefore opening the possibility that the independence requirement may be relaxed [Adams and Nobel, 2010, 2013; Catoni, 2004; van Handel, 2013].

- **Dependency on ε.** If the probability mass of a range R is smaller than ε, then a sample of size as suggested by (2.2) may not contain any point in R. This is especially annoying when a lot of ranges may have very small probability masses. In this case, a lower ε should be used in order to compute acceptable approximations of these probability masses. The obvious drawback in using a lower ε is that it directly corresponds to a larger sample. Given that the sample size depends on $1/\varepsilon^2$, even a small decrease in ε may lead to a substantial increase in the number of samples. A larger sample means a slower algorithm, and the advantages of using sampling may be lost or hindered. Before considering the use of VC-dimension, it is therefore necessary to assess whether we are interested in good approximations of very small probability masses.

Future work

In this dissertation we studied the development and analysis of fast and efficient algorithms for data analytics by taking into consideration different aspects of the problem and different solutions. For each of these, there are a number of possible directions for future research.

- VC-dimension is one of many concepts from the broad field of statistical learning theory. Other tools and results include data-dependent sample complexity bounds based on Rademacher averages, pseudodimension, VC-shatter coefficients, Bayesian uniform bounds, just to name a few [Anthony and Bartlett, 1999; Boucheron et al., 2005; Devroye et al., 1996]. We showed that VC-dimension can be have a practical impact to speed up algorithms for data analytics, but it would be interesting to pursuit the use of these other concepts for the creation of new algorithm or to further reduce the number of samples needed to approximate.

- In this work we focused on "one-shot" sampling, i.e., the algorithms create a single sample with size sufficient to obtain an approximation of desired quality. *Data-dependent bounds based on Rademacher averages* can be useful to develop *progressive sampling* algorithms that start from a small sample size and check a stopping condition to understand whether they sampled enough to obtain a good approximation [Elomaa and Kääriäinen, 2002; Kääriäinen, 2004]. The challenge in creating such algorithms is in the development and analysis of a stopping condition that can detect when to stop as soon as possible.

- A bound on the VC-dimension allows to estimate the probability masses associated to ranges or, equivalently, the expectations of 0-1 functions using their empirical averages. *Pseudodimension* instead works for *real-valued* functions. Since many datasets are real valued or can be seen as collection of real valued functions (e.g., audio and video files and sensor signals), investigating the use of pseudodimension to create sampling-based randomized algorithms for data analytics problems on real data is an interesting research direction.

- One issue that we only partially tackle in Ch. 5 is that of *statistical validation of the results*. We already explained in Ch. 1 why only recently it started to receive attention in data analytics. Although it is of primary importance to go beyond simple multi-hypothesis testing corrections like the Union bound (a.k.a. the Bonferroni inequality) and we achieved it in Ch. 5 for the problem of frequent itemsets, there are other options to control false positives. One example is

controlling the False Discovery Rate [Benjamini and Hochberg, 1995] instead of the probability of having a false positive (which is what we do in Ch. 5 and is known as controlling the Family-Wide Error Rate). There is huge room for improvement and the use of VC-dimension to develop statistical tests to avoid the inclusion of false positives in the results is only one possible directions.

- In Ch. 4 we showed how it is possible to exploit the power and scalability of modern computational platforms like MapReduce to create algorithms for data analytics that can handle huge datasets that could not be processed by a single machine. Technology is one of the driving forces behind the development of new algorithms. It is important to leverage on the properties of the next-generation of computational platforms and of hardware when creating algorithms, in order to achieve the best performances. One particularly interesting direction is the use of parallel/distributed platforms for data streams. Algorithms for data streams have existed for a long time, but they usually consider a single stream processed by a single machine. Platforms like Spark [Zaharia et al., 2010] (see also `http://spark.incubator.apache.org/`), where the stream is partitioned across multiple machines require new algorithms that exploit their power. We implemented a streaming version of PARMA that was integrated in Yahoo SAMOA [De Francisci Morales, 2013]. Adapting other MapReduce algorithms to a distributed streaming setting or creating new algorithms for such platforms is a challenging research direction.

Research in the field of data analytics seems to have paid only limited attention to recent developments in statistics and probability theory. We believe that there may be room for improvements and significant contributions in exploring the literature of these and other fields and adapt results and notions that may have been considered only of theoretical interest to develop efficient algorithm for practical problems involving the analysis of the ever-growing and ever-changing amount of data available today. At the same time, researchers should be aware of advancements in technology and in the computational properties of modern platforms, in order to be able to extract the maximum amount of information from the data as efficiently and fast as possible.

Bibliography

Ittai Abraham, Daniel Delling, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck. VC-dimension and shortest path algorithms. In *Automata, Languages and Programming*, volume 6755 of *Lecture Notes in Computer Science*, pages 690–699. Springer Berlin Heidelberg, 2011. doi: 10.1007/978-3-642-22006-7_58.

Swarup Acharya, Phillip B. Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. Join synopses for approximate query answering. *SIGMOD Rec.*, 28:275–286, June 1999. ISSN 0163-5808. doi: 10.1145/304181.304207.

Terrence M. Adams and Andrew B. Nobel. Uniform convergence of Vapnik–Chervonenkis classes under ergodic sampling. *The Annals of Probability*, 38(4):1345–1367, July 2010. doi: 10.1214/09-AOP511.

Terrence M. Adams and Andrew B. Nobel. *A counterexample concerning the extension of uniform strong laws to ergodic processes*, volume 9 of *Collections*, pages 1–4. Institute of Mathematical Statistics, Beachwood, Ohio, USA, 2013. doi: 10.1214/12-IMSCOLL901.

Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proc. 20th Int. Conf. Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc. ISBN 1-55860-153-8.

Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22:207–216, June 1993. ISSN 0163-5808. doi: 10.1145/170036.170072.

Donald Aingworth, Chandra Chekuri, Piotr Indyk, and Rajeev Motwani. Fast estimation of diameter

and shortest paths (without matrix multiplication). *SIAM J. Comput.*, 28(4):1167–1181, 1999. doi: 10.1137/S0097539796303421.

Noga Alon and Joel H. Spencer. *The Probabilistic Method*. Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, Hoboken, NJ, USA, third edition, 2008.

Jac. M. Anthonisse. The rush in a directed graph. Technical Report BN 9/71, Stichting Mathematisch Centrum, Amsterdam, Netherlands, 1971.

Martin Anthony and Peter L. Bartlett. *Neural Network Learning - Theoretical Foundations*. Cambridge University Press, New York, NY, USA, 1999. ISBN 978-0-521-57353-5.

Ashwin Arulselvan. A note on the set union knapsack problem. *Discrete Applied Mathematics*, 169 (0):214 – 218, 2014. ISSN 0166-218X. doi: 10.1016/j.dam.2013.12.015.

Brian Babcock, Surajit Chaudhuri, and Gautam Das. Dynamic sample selection for approximate query processing. In *Proc. 2003 ACM SIGMOD Int. Conf. Management of Data*, SIGMOD '03, pages 539–550, New York, NY, USA, 2003. ACM. ISBN 1-58113-634-X. doi: 10.1145/872757.872822.

David A. Bader, Shiva Kintali, Kamesh Madduri, and Milena Mihail. Approximating betweenness centrality. In Anthony Bonato and FanR.K. Chung, editors, *Algorithms and Models for the Web-Graph*, volume 4863 of *Lecture Notes in Computer Science*, pages 124–137. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-77003-9. doi: 10.1007/978-3-540-77004-6_10.

Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286 (5439):509–512, 1999.

Michael Benedikt and Leonid Libkin. Aggregate operators in constraint query languages. *J. Comput. Syst. Sci.*, 64(3):628–654, 2002. ISSN 0022-0000. doi: 10.1006/jcss.2001.1810.

Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):289–300, 1995. ISSN 00359246.

Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In *Proc. 40th Annu. ACM Symp. Theory of Computing*, STOC '08, pages 609–618, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-047-0. doi: 10.1145/1374376.1374464.

Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *J. ACM*, 36:929–965, October 1989. ISSN 0004-5411. doi: 10.1145/76359.76371.

Krists Boitmanis, Kārlis Freivalds, Pēteris Ledinš, and Rūdolfs Opmanis. Fast and simple approximation of the diameter and radius of a graph. In Carme Alvarez and Maria Serna, editors, *Experimental Algorithms*, volume 4007 of *Lecture Notes in Computer Science*, pages 98–108. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-34597-8. doi: 10.1007/11764298_9.

Stephen P. Borgatti and Martin G. Everett. A graph-theoretic perspective on centrality. *Soc. Netw.*, 28(4):466–484, 2006. ISSN 0378-8733. doi: 10.1016/j.socnet.2005.11.005.

Stéphane Boucheron, Olivier Bousquet, and Gábor Lugosi. Theory of classification : A survey of some recent advances. *ESAIM: Probability and Statistics*, 9:323–375, 2005.

Ulrik Brandes. A faster algorithm for betweenness centrality. *J. Math. Sociol.*, 25(2):163–177, 2001. doi: 10.1080/0022250X.2001.9990249.

Ulrik Brandes. On variants of shortest-path betweenness centrality and their generic computation. *Soc. Netw.*, 30(2):136–145, 2008. ISSN 0378-8733. doi: 10.1016/j.socnet.2007.11.001.

Ulrik Brandes and Christian Pich. Centrality estimation in large networks. *Int. J. Bifurcation and Chaos*, 17(7):2303–2318, 2007. doi: 10.1142/S0218127407018403.

Hervé Brönnimann, Bin Chen, Manoranjan Dash, Peter Haas, and Peter Scheuermann. Efficient data reduction with ease. In *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Disc. Data Mining*, KDD '03, pages 59–68, New York, NY, USA, 2003. ACM. ISBN 1-58113-737-0. doi: 10.1145/956750.956761.

Paul G. Brown and Peter J. Haas. Techniques for warehousing of sample data. In *Proc. 22nd Int. Conf. Data Eng.*, ICDE '06, page 6, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2570-9. doi: 10.1109/ICDE.2006.157.

Nicolas Bruno and Surajit Chaudhuri. Conditional selectivity for statistics on query expressions. In *Proc. 2004 ACM SIGMOD Int. Conf. Management of Data*, SIGMOD '04, pages 311–322, New York, NY, USA, 2004. ACM. ISBN 1-58113-859-8. doi: 10.1145/1007568.1007604.

Nicolas Bruno, Surajit Chaudhuri, and Luis Gravano. STHoles: a multidimensional workload-aware histogram. *SIGMOD Rec.*, 30:211–222, May 2001. ISSN 0163-5808. doi: 10.1145/376284.375686.

Gregory Buehrer, Srinivasan Parthasarathy, Shirish Tatikonda, Tahsin Kurc, and Joel Saltz. Toward terabyte pattern mining: an architecture-conscious solution. In *Proc. 12th ACM SIGPLAN Symp. Principles and Practice of Parallel Programming*, PPoPP '07, pages 2–12, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-602-8. doi: 10.1145/1229428.1229432.

Toon Calders, Christophe Rigotti, and Jean-François Boulicaut. A survey on condensed representations for frequent sets. In *Constraint-Based Mining and Inductive Databases*, volume 3848 of *Lecture Notes in Computer Science*, pages 64–80. Springer Berlin Heidelberg, 2006. doi: 10.1007/11615576_4.

Olivier Catoni. Improved Vapnik Cervonenkis bounds. *arXiv preprint*, math/0410280, 2004.

Aaron Ceglar and John F. Roddick. Association mining. *ACM Comput. Surv.*, 38:5, July 2006. ISSN 0360-0300. doi: 10.1145/1132956.1132958.

Venkatesan T. Chakaravarthy, Vinayaka Pandit, and Yogish Sabharwal. Analysis of sampling techniques for association rule mining. In *Proc. 12th Int. Conf. Database Theory*, ICDT '09, pages 276–283, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-423-2. doi: 10.1145/1514894.1514927.

Bala Chandra and Shalini Bhaskar. A new approach for generating efficient sample from market basket data. *Expert Sys. with Appl.*, 38(3):1321–1325, 2011. ISSN 0957-4174. doi: 10.1016/j.eswa.2010.07.008.

Surajit Chaudhuri, Rajeev Motwani, and Vivek Narasayya. Random sampling for histogram construction: how much is enough? *SIGMOD Rec.*, 27:436–447, June 1998. ISSN 0163-5808. doi: 10.1145/276305.276343.

Surajit Chaudhuri, Rajeev Motwani, and Vivek Narasayya. On random sampling over joins. In *Proc. 1999 ACM SIGMOD Int. Conf. Management of Data*, SIGMOD '99, pages 263–274, New York, NY, USA, 1999. ACM. ISBN 1-58113-084-8. doi: 10.1145/304182.304206.

Surajit Chaudhuri, Gautam Gas, and Vivek Narasayya. Optimized stratified sampling for approximate query processing. *ACM Trans. Database Syst.*, 32(0):50, June 2007.

Bernard Chazelle. *The discrepancy method: randomness and complexity.* Cambridge University Press, New York, NY, USA, 2000. ISBN 0-521-00357-1.

Bin Chen, Peter Haas, and Peter Scheuermann. A new two-phase sampling based algorithm for discovering association rules. In *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Disc. Data Mining*, KDD '02, pages 462–468, New York, NY, USA, 2002. ACM. ISBN 1-58113-567-X. doi: 10.1145/775047.775114.

Chyouhwa Chen, Shi-Jinn Horng, and Chin-Pin Huang. Locality sensitive hashing for sampling-based algorithms in association rule mining. *Expert Sys. with Appl.*, 38(10):12388–12397, 2011. ISSN 0957-4174. doi: 10.1016/j.eswa.2011.04.018. URL `http://www.sciencedirect.com/science/article/pii/S0957417411005343`.

Meng Chang Chen, Lawrence McNamee, and Norman S. Matloff. Selectivity estimation using homogeneity measurement. In *Proc. 6th Int. Conf. Data Eng.*, ICDE '90, pages 304–310, Washington, DC, USA, 1990. IEEE Computer Society. ISBN 0-8186-2025-0. URL `http://portal.acm.org/citation.cfm?id=645475.654001`.

Yin-Ling Cheung and Ada Wai-Chee Fu. Mining frequent itemsets without support threshold: With and without item constraints. *IEEE Trans. Knowl. Data Eng.*, 16:1052–1069, September 2004. ISSN 1041-4347. doi: 10.1109/TKDE.2004.44.

Flavio Chierichetti, Ravi Kumar, and Andrew Tomkins. Max-cover in Map-Reduce. In *Proc. 19th Int. Conf. World Wide Web*, WWW '10, pages 231–240, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-799-8. doi: 10.1145/1772690.1772715.

Cheng-Tao Chu, Sang Kyun Kim, Yi-An Lin, YuanYuan Yu, Gary R. Bradski, Andrew Y. Ng, and Kunle Olukotun. Map-Reduce for machine learning on multicore. In Bernhard Schölkopf, John C. Platt, and Thomas Hoffman, editors, *Advances in Neural Inform. Proc. Sys. 19, Proc. 20th Annu. Conf. Neural Inform. Process. Sys., Vancouver, British Columbia, Canada, December 4-7, 2006*, pages 281–288. MIT Press, 2007.

Kun-Ta Chuang, Ming-Syan Chen, and Wen-Chieh Yang. Progressive sampling for association rules based on sampling error estimation. In Tu Ho, David Cheung, and Huan Liu, editors, *Advances in Knowl. Disc. and Data Mining*, volume 3518 of *Lecture Notes in Computer Science*, pages 37–44. Springer Berlin Heidelberg, 2005.

Kun-Ta Chuang, Jiun-Long Huang, and Ming-Syan Chen. Power-law relationship and self-similarity

in the itemset support distribution: analysis and applications. *VLDB J.*, 17(5):1121–1141, August 2008. ISSN 1066-8888. doi: 10.1007/s00778-007-0054-1.

Frans Coenen. The LUCS-KDD FP-growth association rule mining algorithm. `http://cgi.csc.liv.ac.uk/~frans/KDD/Software/FPgrowth/fpGrowth.html`, 2014.

Shengnan Cong, Jiawei Han, Jay Hoeflinger, and David Padua. A sampling-based framework for parallel data mining. In *Proc. 10th ACM SIGPLAN Symp. Principles and practice of parallel programming*, PPoPP '05, pages 255–265, New York, NY, USA, 2005. ACM. ISBN 1-59593-080-9. doi: 10.1145/1065944.1065979.

Laurentiu Cristofor. Artool. `http://www.cs.umb.edu/~laur/ARtool/`, 2006.

J.-D. Cryans, S. Ratté, and R. Champagne. Adaptation of APriori to MapReduce to build a warehouse of relations between named entities across the web. In *2nd Int. Conf. Advances in Databases Knowl. Data Applications (DBKDA), 2010*, pages 185–189, April 2010. doi: 10.1109/DBKDA.2010.34.

Gábor Csárdi and Tamás Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006. URL `http://igraph.sf.net`.

Gautam Das. Sampling methods in approximate query answering systems. In John Wang, editor, *Encyclopedia of Data Warehousing and Mining*, pages 1702–1707. IGI Global, Hershey, PA, USA, 2nd edition, 2009.

Gianmarco De Francisci Morales. SAMOA: a platform for mining big data streams. In Leslie Carr, Alberto H. F. Laender, Bernadette Farias Lóscio, Irwin King, Marcus Fontoura, Denny Vrandecic, Lora Aroyo, José Palazzo M. de Oliveira, Fernanda Lima, and Erik Wilde, editors, *WWW (Companion Volume)*, pages 777–778. International World Wide Web Conferences Steering Committee / ACM, 2013. ISBN 978-1-4503-2038-2.

Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. *C. ACM*, 51(1):107–113, 2008.

Luc Devroye, László Györfi, and Gábor Lugosi. *A Probabilistic Theory of Pattern Recognition*. Applications of Mathematics. Springer Berlin Heidelberg, 1996. ISBN 9780387946184.

Alin Dobra. Histograms revisited: when are histograms the best approximation method for aggregates over joins? In *Proc. 24th ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems*, PODS '05, pages 228–237, New York, NY, USA, 2005. ACM. ISBN 1-59593-062-0. doi: 10.1145/1065167.1065196.

Shlomi Dolev, Yuval Elovici, and Rami Puzis. Routing betweenness centrality. *J. ACM*, 57(4): 25:1–25:27, May 2010. ISSN 0004-5411. doi: 10.1145/1734213.1734219.

Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009. ISBN 978-0-521-88427-3.

William DuMouchel and Daryl Pregibon. Empirical Bayes screening for multi-item associations. In *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Disc. Data Mining*, KDD '01, pages 67–76, New York, NY, USA, 2001. ACM. ISBN 1-58113-391-X. doi: 10.1145/502512.502526.

Mohammad El-hajj and Osmar R. Zaïane. Parallel leap: large-scale maximal pattern mining in a distributed environment. In *12th Int. Conf. Parallel and Distributed Systems*, volume 1 of *ICPADS '06*, page 8 pp., 2006. doi: 10.1109/ICPADS.2006.77.

Tapio Elomaa and Matti Kääriäinen. Progressive rademacher sampling. In Rina Dechter and Richard S. Sutton, editors, *AAAI/IAAI*, pages 140–145. AAAI Press / The MIT Press, 2002.

David Eppstein and Joseph Wang. Fast approximation of centrality. *J. Graph Algorithms Appl.*, 8 (1):39–45, 2004.

Cristian Estan and Jeffrey F. Naughton. End-biased samples for join cardinality estimation. In *Proc. 22nd Int. Conf. Data Eng*, ICDE '06, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2570-9. doi: 10.1109/ICDE.2006.61.

W. Fang, K. K. Lau, M. Lu, X. Xiao, C. K. Lam, Y. Yang, B. He, Q. Luo, P. V. Sander, and K. Yang. Parallel data mining on graphics processors. Technical Report 07, The Hong Kong University of Science & Technology, 2008.

Uriel Feige and Mohammad Mahdian. Finding small balanced separators. In *Proc. 38th ann. ACM Symp. Theory of Computing*, STOC '06, pages 375–384, New York, NY, USA, 2006. ACM. ISBN 1-59593-134-1. doi: 10.1145/1132516.1132573.

Lester R. Ford and Delbert R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40:35–41, 1977.

Ada Wai-Chee Fu, Renfrew W.-w. Kwong, and Jian Tang. Mining n-most interesting itemsets. In *Proc. 12th Int. Symp. Foundations of Intelligent Systems*, ISMIS '00, pages 59–67. Springer Berlin Heidelberg, 2000. ISBN 3-540-41094-5.

Sorabh Gandhi, Subhash Suri, and Emo Welzl. Catching elephants with mice: Sparse sampling for monitoring sensor networks. *ACM Trans. Sensor Netw.*, 6(1):1:1–1:27, January 2010. ISSN 1550-4859. doi: 10.1145/1653760.1653761.

Sumit Ganguly, Phillip B. Gibbons, Yossi Matias, and Avi Silberschatz. Bifocal sampling for skew-resistant join size estimation. *SIGMOD Rec.*, 25:271–281, June 1996. ISSN 0163-5808. doi: 10.1145/235968.233340.

Venkatesh Ganti, Mong-Li Lee, and Raghu Ramakrishnan. ICICLES: Self-tuning samples for approximate query answering. In *Proc. 26th int. Conf. Very Large Data Bases*, VLDB '00, pages 176–187, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1-55860-715-3.

Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. *Database Systems - The Complete Book*. Prentice Hall, Upper Saddle River, NJ, USA, 2002. ISBN 978-0-13-187325-4.

Robert Geisberger, Peter Sanders, and Dominik Schultes. Better approximation of betweenness centrality. In J. Ian Munro and Dorothea Wagner, editors, *Algorithm Eng. & Experiments (ALENEX'08)*, pages 90–100. SIAM, 2008.

Rainer Gemulla, Wolfgang Lehner, and Peter J. Haas. A dip in the reservoir: maintaining sample synopses of evolving datasets. In *Proc. 32nd Int. Conf. Very Large Data Bases*, VLDB '06, pages 595–606, Almaden, CA, USA, 2006. VLDB Endowment.

Rainer Gemulla, Wolfgang Lehner, and Peter J. Haas. Maintaining Bernoulli samples over evolving multisets. In *Proc. 26th ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems*, PODS '07, pages 93–102, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-685-1. doi: 10.1145/1265530.1265544.

Liqiang Geng and Howard J. Hamilton. Interestingness measures for data mining: A survey. *ACM Comput. Surv.*, 38(3), September 2006. ISSN 0360-0300. doi: 10.1145/1132960.1132963.

Lise Getoor, Benjamin Taskar, and Daphne Koller. Selectivity estimation using probabilistic models. *SIGMOD Rec.*, 30:461–472, May 2001. ISSN 0163-5808. doi: 10.1145/376284.375727.

Amol Ghoting, Prabhanjan Kambadur, Edwin Pednault, and Ramakrishnan Kannan. NIMBLE: a toolkit for the implementation of parallel data mining and machine learning algorithms on mapreduce. In *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Disc. Data Mining*, KDD '11, pages 334–342, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0813-7. doi: 10.1145/2020408.2020464.

Phillip B. Gibbons and Yossi Matias. New sampling-based summary statistics for improving approximate query answers. *SIGMOD Rec.*, 27:331–342, June 1998. ISSN 0163-5808. doi: 10.1145/276305.276334.

Aristides Gionis, Heikki Mannila, Taneli Mielikäinen, and Panayiotis Tsaparas. Assessing data mining results via swap randomization. *ACM Trans. Knowl. Disc. from Data*, 1(3), December 2007. ISSN 1556-4681. doi: 10.1145/1297332.1297338.

Bart Goethals and Mohammed J. Zaki. Advances in frequent itemset mining implementations: report on FIMI'03. *SIGKDD Explor. Newsl.*, 6(1):109–117, June 2004. ISSN 1931-0145. doi: 10.1145/1007730.1007744.

Olivier Goldschmidt, David Nehme, and Gang Yu. Note: On the set-union knapsack problem. *Naval Research Logistics*, 41(6):833–842, 1994.

Michael T. Goodrich, Nodari Sitchinava, and Qin Zhang. Sorting, searching, and simulation in the MapReduce framework. *CoRR*, abs/1101.1902, 2011.

Gösta Grahne and Jianfei Zhu. Efficiently using prefix-trees in mining frequent itemsets. In Bart Goethals and Mohammed Javeed Zaki, editors, *FIMI*, volume 90 of *CEUR Workshop Proc.* CEUR-WS.org, 2003.

David Gross-Amblard. Query-preserving watermarking of relational databases and xml documents. *ACM Trans. Database Syst.*, 36:3:1–3:24, March 2011. ISSN 0362-5915. doi: 10.1145/1929934.1929937.

Jarek Gryz and Dongming Liang. Query selectivity estimation via data mining. In Mieczyslaw A. Klopotek, Slawomir T. Wierzchon, and Krzysztof Trojanowski, editors, *Intelligent Information Processing and Web Mining, Proc. Int. IIS: IIPWM'04 Conf .held in Zakopane, Poland, May 17-20, 2004*, Advances in Soft Computing, pages 29–38. Springer Berlin / Heidelberg, 2004.

Peter J. Haas. Hoeffding inequalities for join-selectivity estimation and online aggregation. Technical Report RJ 10040, IBM Almaden Research, 1996.

Peter J. Haas and Christian König. A bi-level bernoulli scheme for database sampling. In *Proc. 2004 ACM SIGMOD Int. Conf. Management of Data*, SIGMOD '04, pages 275–286, New York, NY, USA, 2004. ACM. ISBN 1-58113-859-8. doi: 10.1145/1007568.1007601.

Peter J. Haas and Arun N. Swami. Sequential sampling procedures for query size estimation. In *Proc. 1992 ACM SIGMOD Int. Conf. Management of Data*, SIGMOD '92, pages 341–350, New York, NY, USA, 1992. ACM. ISBN 0-89791-521-6. doi: 10.1145/130283.130335.

Peter J. Haas and Arun N. Swami. Sampling-based selectivity estimation for joins using augmented frequent value statistics. In *Proc. 11th Int. Conf. Data Eng.*, ICDE '95, pages 522–531, Washington, DC, USA, 1995. IEEE Computer Society. ISBN 0-8186-6910-1.

Peter J. Haas, Jeffrey F. Naughton, S. Seshadri, and Arun N. Swami. Fixed-precision estimation of join selectivity. In *Proc. 12th ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems*, PODS '93, pages 190–201, New York, NY, USA, 1993. ACM. ISBN 0-89791-593-3. doi: 10.1145/153850.153875.

Peter J. Haas, Jeffrey F. Naughton, and Arun N. Swami. On the relative cost of sampling for join selectivity estimation. In *Proc. 13th ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems*, PODS '94, pages 14–24, New York, NY, USA, 1994. ACM. ISBN 0-89791-642-5. doi: 10.1145/182591.182594.

Peter J. Haas, Jeffrey F. Naughton, S. Seshadri, and Arun N. Swami. Selectivity and cost estimation for joins based on random sampling. *J. Comput. Syst. Sci.*, 52:550–569, June 1996. ISSN 0022-0000. doi: 10.1006/jcss.1996.0041.

M. T. Hajiaghayi, K. Jain, K. Konwar, L. C. Lau, I. I. Măndoiu, A. Russell, A. Shvartsman, and V. V.

Vazirani. The minimum k-colored subgraph problem in haplotyping and dna primer selection. In *Proc. Int. Workshop on Bioniformatics Research and Applications (IWBRA)*, 2006.

Wilhelmiina Hämäläinen. StatApriori: an efficient algorithm for searching statistically significant association rules. *Knowl. Inf. Sys.*, 23(3):373–399, 2010. doi: 10.1007/s10115-009-0229-8.

Suhel Hammoud. *MapReduce Network Enabled Algorithms for Classification Based on Association Rules*. PhD thesis, Brunel University, 2011.

Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In Weidong Chen, Jeffrey F. Naughton, and Philip A. Bernstein, editors, *SIGMOD Conf.*, pages 1–12. ACM, 2000. ISBN 1-58113-218-2.

Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. Frequent pattern mining: current status and future directions. *Data Mining and Knowl. Disc.* , 15:55–86, 2007. ISSN 1384-5810. doi: 10.1007/s10618-006-0059-1.

Sami Hanhijärvi. Multiple hypothesis testing in pattern discovery. In *Discovery Science*, volume 6926 of *Lecture Notes in Computer Science*, pages 122–134. Springer Berlin Heidelberg, 2011. doi: 10.1007/978-3-642-24477-3_12.

Sariel Har-Peled and Micha Sharir. Relative (p, ε)-approximations in geometry. *Discrete & Computational Geometry*, 45(3):462–496, 2011. ISSN 0179-5376. doi: 10.1007/s00454-010-9248-1.

Banchong Harangsri, John Shepherd, and Anne H. H. Ngu. Query size estimation using machine learning. In *Proc. 5th Int. Conf. Database Systems for Advanced Applications (DASFAA)*, pages 97–106, Hackensack, NJ, USA, 1997. World Scientific Press. ISBN 981-02-3107-5.

Trevor Hastie, Robert Tibshirani, and Jeroem Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer-Verlag, 2009.

David Haussler and Emo Welzl. Epsilon-nets and simplex range queries. In *Proc. 2nd Annu. Symp. Computational geometry*, SCG '86, pages 61–71, New York, NY, USA, 1986. ACM. ISBN 0-89791-194-6. doi: 10.1145/10515.10522.

Ruefei He and Jonathan Shapiro. Bayesian mixture models for frequent itemset discovery. *CoRR*, abs/1209.6001:1–33, September 2012.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *J. American Statistical Assoc.*, 58(301):13–30, 1963.

John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973.

Wen-Chi Hou, Gultekin Ozsoyoglu, and Baldeo K. Taneja. Statistical estimators for relational algebra expressions. In *Proc. 7th ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems*, PODS '88, pages 276–287, New York, NY, USA, 1988. ACM. ISBN 0-89791-263-2. doi: 10.1145/308386.308455.

Wen-Chi Hou, Gultekin Ozsoyoglu, and Erdogan Dogdu. Error-constrained count query evaluation in relational databases. *SIGMOD Rec.*, 20:278–287, April 1991. ISSN 0163-5808. doi: 10.1145/119995.115837.

Xuegang Hu and Haitao Yu. The research of sampling for mining frequent itemsets. In Guo-Ying Wang, James Peters, Andrzej Skowron, and Yiyu Yao, editors, *Rough Sets and Knowl. Tech.*, volume 4062 of *Lecture Notes in Computer Science*, pages 496–501. Springer Berlin Heidelberg, 2006. doi: 10.1007/11795131_72.

Wontae Hwang and Dongseung Kim. Improved association rule mining by modified trimming. In *Proc. 6th IEEE Int. Conf. Computer and Inform. Tech.*, CIT '06, page 24. IEEE Computer Society, September 2006. doi: 10.1109/CIT.2006.101.

Yannis E. Ioannidis and Viswanath Poosala. Balancing histogram optimality and practicality for query result size estimation. In *Proc. 1995 ACM SIGMOD Int. Conf. Management of Data*, SIGMOD '95, pages 233–244, New York, NY, USA, 1995. ACM. ISBN 0-89791-731-6. doi: 10.1145/223784.223841.

Riko Jacob, Dirk Koschützki, KatharinaAnna Lehmann, Leon Peeters, and Dagmar Tenfelde-Podehl. Algorithms for centrality indices. In Ulrik Brandes and Thomas Erlebach, editors, *Network Analysis*, volume 3418 of *Lecture Notes in Computer Science*, pages 62–82. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-24979-5. doi: 10.1007/978-3-540-31955-9_4. URL `http://dx.doi.org/10.1007/978-3-540-31955-9_4`.

H. V. Jagadish, Nick Koudas, S. Muthukrishnan, Viswanath Poosala, Kenneth C. Sevcik, and Torsten Suel. Optimal histograms with quality guarantees. In *Proc. 24th int. Conf. Very Large Data Bases*, VLDB '98, pages 275–286, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1-55860-566-5.

Christopher Jermaine, Abhijit Pol, and Subramanian Arumugam. Online maintenance of very large random samples. In *Proc. 2004 ACM SIGMOD Int. Conf. Management of Data*, SIGMOD '04, pages 299–310, New York, NY, USA, 2004. ACM. ISBN 1-58113-859-8. doi: 10.1145/1007568.1007603.

Cai-Yan Jia and Xie-Ping Gao. Multi-scaling sampling: An adaptive sampling method for discovering approximate association rules. *J. Comp. Sci. Techn.*, 20:309–318, 2005. ISSN 1000-9000. doi: 10.1007/s11390-005-0309-5.

Caiyan Jia and Ruqian Lu. Sampling ensembles for frequent patterns. In Lipo Wang and Yaochu Jin, editors, *Fuzzy Systems and Knowl. Disc.* , volume 3613 of *Lecture Notes in Computer Science*, pages 478–478. Springer Berlin Heidelberg, 2005. doi: 10.1007/11539506_150.

Ruoming Jin, Leo Glimcher, Chris Jermaine, and Gagan Agrawal. New sampling-based estimators for olap queries. In *Proc. 22nd Int. Conf. Data Eng.*, ICDE '06, page 18, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2570-9. doi: 10.1109/ICDE.2006.106.

George H. John and Pat Langley. Static versus dynamic sampling for data mining. In *Proc. 2nd Int. Conf. Knowl. Disc. Data Mining*, KDD '96, pages 367–370, Menlo Park, CA, USA, 1996. The AAAI Press.

Shantanu Joshi and Christopher Jermaine. Robust stratified sampling plans for low selectivity queries. In *Proc. 2008 IEEE 24th Int. Conf. Data Eng.*, pages 199–208, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-1-4244-1836-7. doi: 10.1109/ICDE.2008.4497428.

Matti Kääriäinen. Relating the Rademacher and VC bounds. Technical Report C-2004-57, Department of Computer Science University of Helsinki, 2004.

Hermann Kaindl and Gerhard Kainz. Bidirectional heuristic search reconsidered. *J. Artif. Intell. Res. (JAIR)*, 7:283–317, 1997. doi: 10.1613/jair.460.

Raghav Kaushik, Jeffrey F. Naughton, Raghu Ramakrishnan, and Venkatesan T. Chakravarthy. Synopses for query optimization: A space-complexity perspective. *ACM Trans. Database Syst.*, 30:1102–1127, December 2005. ISSN 0362-5915. doi: 10.1145/1114244.1114251.

Michael J. Kearns and Umesh Virkumar Vazirani. *An introduction to computational learning theory.* MIT press, 1994.

Adam Kirsch, Michael Mitzenmacher, Andrea Pietracaprina, Geppino Pucci, Eli Upfal, and Fabio Vandin. An efficient rigorous approach for identifying statistically significant frequent itemsets. *J. ACM*, 59(3):12:1–12:22, June 2012. ISSN 0004-5411. doi: 10.1145/2220357.2220359.

Jon M. Kleinberg. Detecting a network failure. *Internet Mathematics*, 1(1):37–55, 2003. doi: 10.1080/15427951.2004.10129077.

Jon M. Kleinberg, Mark Sandler, and Aleksandrs Slivkins. Network failure detection and graph connectivity. *SIAM J. Comput.*, 38(4):1330–1346, 2008. doi: 10.1137/070697793.

Nicolas Kourtellis, Tharaka Alahakoon, Ramanuja Simha, Adriana Iamnitchi, and Rahul Tripathi. Identifying high betweenness centrality nodes in large social networks. *Soc. Netw. Anal. and Mining*, pages 1–16, 2012. ISSN 1869-5450. doi: 10.1007/s13278-012-0076-6.

Evangelos Kranakis, Danny Krizanc, Berthold Ruf, Jorge Urrutia, and Gerhard Woeginger. The VC-dimension of set systems defined by graphs. *Discrete Applied Mathematics*, 77(3):237–257, 1997. ISSN 0166-218X. doi: 10.1016/S0166-218X(96)00137-0.

Per-Ake Larson, Wolfgang Lehner, Jingren Zhou, and Peter Zabback. Cardinality estimation using sample views with quality assurance. In *Proc. 2007 ACM SIGMOD Int. Conf. Management of Data*, SIGMOD '07, pages 175–186, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-686-8. doi: 10.1145/1247480.1247502.

Erich L. Lehmann and George Casella. *Theory of Point Estimation.* Springer-Verlag New York, Inc., 1998.

Haoyuan Li, Yi Wang, Dong Zhang, Ming Zhang, and Edward Y. Chang. PFP: Parallel FP-Growth for query recommendation. In *Proc. 2008 ACM Conf. Recommender Sys.*, RecSys '08, pages 107–114, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-093-7. doi: 10.1145/1454008.1454027.

Lingjuan Li and Min Zhang. The strategy of mining association rule based on cloud computing. In *Int. Conf. Business Computing and Global Informatization (BCGIN), 2011*, pages 475–478, July 2011. doi: 10.1109/BCGIn.2011.125.

Yanrong Li and Raj Gopalan. Effective sampling for mining association rules. In Geoffrey Webb and Xinghuo Yu, editors, *AI 2004: Advances in Artificial Intelligence*, volume 3339 of *Lecture Notes in Computer Science*, pages 73–75. Springer Berlin Heidelberg, 2005. doi: 10.1007/978-3-540-30549-1_35.

Yi Li, Philip M. Long, and Aravind Srinivasan. Improved bounds on the sample complexity of learning. *J. Comp. Sys. Sci.*, 62(3):516–527, 2001. ISSN 0022-0000. doi: 10.1006/jcss.2000.1741.

Yeon-sup Lim, Daniel S. Menasche, Bruno Ribeiro, Don Towsley, and Prithwish Basu. Online estimating the k central nodes of a network. In *IEEE Network Science Workshop*, NSW'11, pages 118–122, 2011. doi: 10.1109/NSW.2011.6004633.

Jimmy Lin and Michael Schatz. Design patterns for efficient graph algorithms in MapReduce. In *Proc. 8th Workshop on Mining and Learning with Graphs*, MLG '10, pages 78–85, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0214-2. doi: 10.1145/1830252.1830263.

Nathan Linial, Yishay Mansour, and Ronald L. Rivest. Results on learnability and the Vapnik-Chervonenkis dimension. *Information and Computation*, 90(1):33–49, 1991. ISSN 0890-5401. doi: 10.1016/0890-5401(91)90058-A.

Richard J. Lipton and Jeffrey F. Naughton. Query size estimation by adaptive sampling. *J. Comput. Syst. Sci.*, 51:18–25, August 1995. ISSN 0022-0000. doi: 10.1006/jcss.1995.1050.

Richard J. Lipton, Jeffrey F. Naughton, and Donovan A. Schneider. Practical selectivity estimation through adaptive sampling. *SIGMOD Rec.*, 19:1–11, May 1990. ISSN 0163-5808. doi: 10.1145/93605.93611.

Guimei Liu, Haojun Zhang, and Limsoon Wong. Controlling false positives in association rule mining. *Proc. VLDB Endow.*, 5(2):145–156, October 2011. ISSN 2150-8097.

Li Liu, Eric Li, Yimin Zhang, and Zhizhong Tang. Optimization of frequent itemset mining on multiple-core processor. In *Proc. 33rd Int. Conf. Very Large Data Bases*, VLDB '07, pages 1275–1285. VLDB Endowment, 2007. ISBN 978-1-59593-649-3.

Maarten Löffler and Jeff M. Phillips. Shape fitting on point sets with probability distributions. In Amos Fiat and Peter Sanders, editors, *Algorithms - ESA 2009*, volume 5757 of *Lecture Notes in Computer Science*, pages 313–324. Springer Berlin Heidelberg, 2009. doi: 10.1007/978-3-642-04128-0_29.

Cécile Low-Kam, Chedy Raïssi, Mehdi Kaytoue, Jian Pei, et al. Mining statistically significant sequential patterns. In *IEEE Int. Conf. Data Mining*, ICDM '13, 2013.

Basel A. Mahafzah, Amer F. Al-Badarneh, and Mohammed Z. Zakaria. A new sampling technique for association rule mining. *J. Inf. Sci.*, 35(3):358–376, 2009. doi: 10.1177/0165551508100382.

Arun S. Maiya and Tanya Y. Berger-Wolf. Online sampling of high centrality individuals in social networks. In MohammedJ. Zaki, JeffreyXu Yu, B. Ravindran, and Vikram Pudi, editors, *Advances in Knowl. Disc. Data Mining*, volume 6118 of *Lecture Notes in Computer Science*, pages 91–98. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-13656-6. doi: 10.1007/978-3-642-13657-3_12.

Michael Mampaey, Nikolaj Tatti, and Jilles Vreeken. Tell me what I need to know: succinctly summarizing data with itemsets. In *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Disc. Data Mining*, KDD '11, pages 573–581, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0813-7. doi: 10.1145/2020408.2020499.

Heikki Mannila, Hannu Toivonen, and Inkeri Verkamo. Efficient algorithms for discovering association rules. In *KDD Workshop*, pages 181–192, Menlo Park, CA, USA, 1994. The AAAI Press.

Volker Markl, Peter. J. Haas, Marcel Kutsch, Nimrod Megiddo, Utkarsh Srivastava, and Tam Minh Tran. Consistent selectivity estimation via maximum entropy. *VLDB J.*, 16:55–76, January 2007. ISSN 1066-8888. doi: 10.1007/s00778-006-0030-1.

Yossi Matias, Jeffrey Scott Vitter, and Min Wang. Wavelet-based histograms for selectivity estimation. In *Proc. 1998 ACM SIGMOD Int. Conf. Management of Data*, SIGMOD '98, pages 448–459, New York, NY, USA, 1998. ACM. ISBN 0-89791-995-5. doi: 10.1145/276304.276344.

Jiří Matoušek. *Lectures on Discrete Geometry*. Springer-Verlag, Secaucus, NJ, USA, 2002. ISBN 0387953744.

Daniel J. McDonald, Cosha Rohilla Shalizi, and Mark Schervish. Estimated VC dimension for risk bounds. *CoRR*, abs/1111.3404, November 2011.

Nimrod Megiddo and Ramakrishnan Srikant. Discovering predictive association rules. In Rakesh Agrawal, Paul E. Stolorz, and Gregory Piatetsky-Shapiro, editors, *Proc. 4th Int. Conf. Knowl. Disc. Data Mining*, KDD '98, pages 274–278, New York, NY, USA, 1998. AAAI Press.

Rupert G. Miller. *Simultaneous Statistical Inference*. Springer series in statistics. Springer-Verlag, New York, NY, USA, 1981.

Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. The MIT Press, 2012.

Elchanan Mossel and Christopher Umans. On the complexity of approximating the VC dimension. In *16th Annu. IEEE Conf. Computational Complexity*, CCC '01, pages 220–225, 2001. doi: 10.1109/CCC.2001.933889.

Thomas Natschläger and Michael Schmitt. Exact VC-dimension of boolean monomials. *Inform. Process. Lett.*, 59(1):19–20, 1996. ISSN 0020-0190. doi: 10.1016/0020-0190(96)00084-1.

Ilan Newman and Yuri Rabinovich. On multiplicative λ-approximations and some geometric applications. In *Proc. 23rd Annu. ACM-SIAM Symp. Discrete Algorithms*, SODA '12, pages 51–67. SIAM, 2012.

Mark E. J. Newman. *Networks – An Introduction*. Oxford University Press, 2010.

Mark E. J. Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, February 2004. doi: 10.1103/PhysRevE.69.026113.

Anne H. H. Ngu, Banchong Harangsri, and John Shepherd. Query size estimation for joins using systematic sampling. *Distrib. Parallel Databases*, 15:237–275, May 2004. ISSN 0926-8782. doi: 10.1023/B:DAPD.0000018573.35050.25.

Frank Olken. *Random Sampling from Databases*. PhD thesis, University of California, Berkeley, 1993.

Tore Opsahl, Filip Agneessens, and John Skvoretz. Node centrality in weighted networks: Generalizing degree and shortest paths. *Soc. Netw.*, 32(3):245–251, 2010. doi: 10.1016/j.socnet.2010.03.006.

Eray Özkural, Bora Uçar, and Cevdet Aykanat. Parallel frequent item set mining with selective item replication. *IEEE Trans. Paralel Distrib. Syst.*, 22(10):1632–1640, October 2011. ISSN 1045-9219. doi: 10.1109/TPDS.2011.32.

Christos H. Papadimitriou and Mihalis Yannakakis. On limited nondeterminism and the complexity of the V-C dimension. *J. Comput. Syst. Sci.*, 53(2):161–170, 1996. doi: 10.1006/jcss.1996.0058.

Srinivasan Parthasarathy. Efficient progressive sampling for association rules. In *Proc. 2002 IEEE Int. Conf. Data Mining*, ICDM '02, pages 354–361. IEEE Computer Society, 2002. doi: 10.1109/ICDM.2002.1183923.

Jürgen Pfeffer and Kathleen M. Carley. k-centralities: local approximations of global measures based on shortest paths. In *Proc. 21st Int. Conf. Companion on World Wide Web*, WWW '12 Companion, pages 1043–1050, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1230-1. doi: 10.1145/2187980.2188239.

Andrea Pietracaprina and Fabio Vandin. Efficient incremental mining of top-K frequent closed itemsets. In Vincent Corruble, Masayuki Takeda, and Einoshin Suzuki, editors, *Discovery Science*, volume 4755 of *Lecture Notes in Computer Science*, pages 275–280. Springer Berlin Heidelberg, 2007. doi: 10.1007/978-3-540-75488-6_29.

Andrea Pietracaprina, Matteo Riondato, Eli Upfal, and Fabio Vandin. Mining top-k frequent itemsets through progressive sampling. *Data Mining Knowl. Disc.*, 21(2):310–326, 2010.

Andrea Pietracaprina, Geppino Pucci, Matteo Riondato, Francesco Silvestri, and Eli Upfal. Space-round tradeoffs for MapReduce computationseduce computations. In Utpal Banerjee, Kyle A. Gallivan, Gianfranco Bilardi, and Manolis Katevenis, editors, *Int. Conf. Supercomputing, ICS'12, Venice, Italy, June 25-29, 2012*, pages 235–244. ACM, 2012. ISBN 978-1-4503-1316-2.

Ira Pohl. *Bidirectional Heuristic Search in Path Problems.* PhD thesis, Stanford University, 1969.

Viswanath Poosala and Yannis E. Ioannidis. Selectivity estimation without the attribute value independence assumption. In *Proc. 23rd Int. Conf. Very Large Data Bases*, VLDB '97, pages 486–495, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. ISBN 1-55860-470-7.

Viswanath Poosala, Peter J. Haas, Yannis E. Ioannidis, and Eugene J. Shekita. Improved histograms for selectivity estimation of range predicates. In *Proc. 1996 ACM SIGMOD Int. Conf. Management of Data*, SIGMOD '96, pages 294–305, New York, NY, USA, 1996. ACM. ISBN 0-89791-794-4. doi: 10.1145/233269.233342.

Christopher Ré and Dan Suciu. Understanding cardinality estimation using entropy maximization. In *Proc. 29th ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems*, PODS '10, pages 53–64, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0033-9. doi: 10.1145/1807085.1807095.

Matteo Riondato and Evgenios M. Kornaropoulos. Fast approximation of betweenness centrality through sampling. In Ben Carterette, Fernando Diaz, Carlos Castillo, and Donald Metzler, editors, *WSDM*, pages 413–422. ACM, 2014. ISBN 978-1-4503-2351-2.

Matteo Riondato and Eli Upfal. Efficient discovery of association rules and frequent itemsets through sampling with tight performance guarantees. *ACM Trans. on Knowl. Disc. from Data*, 2014.

Matteo Riondato and Fabio Vandin. Finding the true frequent itemsets. In *Proc. SIAM Int. Data Mining Conf.*, 2014.

Matteo Riondato, Mert Akdere, Uğur Çetintemel, Stanley B. Zdonik, and Eli Upfal. The VC-dimension of SQL queries and selectivity estimation through sampling. In Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba, and Michalis Vazirgiannis, editors, *Mach. Learn. Knowl. Disc. Databases - European Conf., ECML PKDD 2011, Athens, Greece, September 5-9, 2011. Proc., Part II*, volume 6912 of *Lecture Notes in Computer Science*, pages 661–676, Berlin / Heidelberg, 2011. Springer. ISBN 978-3-642-23782-9.

Matteo Riondato, Justin A. DeBrabant, Rodrigo Fonseca, and Eli Upfal. PARMA: A parallel randomized algorithm for association rules mining in MapReduce. In Xue-wen Chen, Guy Lebanon, Haixun Wang, and Mohammed J. Zaki, editors, *CIKM '12: Proc. 21st ACM Int. Conf. Inform. Knowl. management*, pages 85–94, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1156-4.

Liam Roditty and Virginia Vassilevska Williams. Approximating the diameter of a graph. *CoRR abs/1207.3622*, July 2012. URL `http://arxiv.org/abs/1207.3622`.

Jin Ruoming, Yang Ge, and Gagan Agrawal. Shared memory parallelization of data mining algorithms: techniques, programming interface, and performance. *IEEE Trans. Knowl. Data Eng.*, 17 (1):71–89, January 2005. ISSN 1041-4347. doi: 10.1109/TKDE.2005.18.

Nick V. Sahinidis and Mohit Tawarmalani. *BARON 9.0.4: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual, 2010.

Ahmet Erdem Sarıyüce, Erik Saule, Kamer Kaya, and Ümit V. Çatalyürek. Shattering and compressing networks for betweenness centrality. In *SIAM Data Mining Conf.*, 2013.

Marcus Schäfer. Deciding the Vapnik-Červonenkis dimension is Σ_3^{p}-complete. *J. Comput. Syst. Sci.*, 58(1):177–182, 1999. doi: 10.1006/jcss.1998.1602.

Tobias Scheffer and Stefan Wrobel. Finding the most interesting patterns in a database quickly by using sequential sampling. *J. Mach. Learn. Res.*, 3:833–862, December 2002. ISSN 1532-4435.

Xuhui Shao, Vladimir Cherkassky, and William Li. Measuring the VC-dimension using optimized experimental design. *Neural Computation*, 12(8):1969–1986, 2000.

Craig Silverstein, Sergey Brin, and Rajeev Motwani. Beyond market baskets: Generalizing association rules to dependence rules. *Data Mining Knowl. Disc.*, 2(1):39–68, January 1998. ISSN 1384-5810. doi: 10.1023/A:1009713703947.

U. Srivastava, Peter J. Haas, V. Markl, M. Kutsch, and T. M. Tran. ISOMER: Consistent histogram construction using query feedback. In *Proc. 22nd Int. Conf. Data Eng.*, ICDE '06, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2570-9. doi: 10.1109/ICDE.2006.84.

Pan-Ning Tan, Vipin Kumar, and Jaideep Srivastava. Selecting the right objective measure for association analysis. *Inf. Sys.*, 29:293–313, 2004.

Hannu Toivonen. Sampling large databases for association rules. In *Proc. 22nd Int. Conf. Very Large Data Bases*, VLDB '96, pages 134–145, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc. ISBN 1-55860-382-4.

Ramon van Handel. The universal Glivenko-Cantelli property. *Probability Theory and Related Fields*, 155(3-4):911–934, 2013.

Vladimir N. Vapnik. *Statistical learning theory.* Wiley, 1998. ISBN 0471030031.

Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Statistics for engineering and information science. Springer-Verlag, New York, NY, USA, 1999. ISBN 9780387987804.

Vladimir N. Vapnik and Alexey J. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971. doi: 10.1137/1116025.

Vladimir N. Vapnik, Esther Levin, and Yan Le Cun. Measuring the VC-dimension of a learning machine. *Neural Computation*, 6:851–876, 1994.

Dinkar Vasudevan and Milan Vojonović. Ranking through random sampling. MSR-TR-2009-8 8, Microsoft Research, 2009.

Jeffrey Scott Vitter. An efficient algorithm for sequential random sampling. *ACM Trans. Math. Softw.*, 13(1):58–67, March 1987. ISSN 0098-3500. doi: 10.1145/23002.23003.

Hai Wang and Kenneth C. Sevcik. A multi-dimensional histogram for selectivity estimation and fast approximate query answering. In *Proc. 2003 Conf. Centre for Advanced Studies on Collaborative research*, CASCON '03, pages 328–342, Boston, MA, USA, 2003. IBM Press.

Jianyong Wang, Jiawei Han, Ying Lu, and Petre Tzvetkov. TFP: An efficient algorithm for mining top-k frequent closed itemsets. *IEEE Trans. Knowl. Data Eng.*, 17:652–664, May 2005a. ISSN 1041-4347. doi: 10.1109/TKDE.2005.81.

Min Wang, Jeffrey Scott Vitter, and Balakrishna R. Iyer. Selectivity estimation in the presence of alphanumeric correlations. In *Proc. 13th Int. Conf. Data Eng.*, ICDE '97, pages 169–180, Washington, DC, USA, 1997. IEEE Computer Society. ISBN 0-8186-7807-0.

Surong Wang, Manoranjan Dash, and Liang-Tien Chia. Efficient sampling: Application to image data. In Tu Bao Ho, David Wai-Lok Cheung, and Huan Liu, editors, *Proc. 9th Pacific-Asia Conf. Advances Knowl. Disc. Data Mining, PAKDD 2005*, volume 3518 of *Lecture Notes in Computer Science*, pages 452–463. Springer Berlin Heidelberg, 2005b.

Geoffrey I. Webb. Discovering significant patterns. *Mach. Learn.*, 68(1):1–33, 2007. doi: 10.1007/s10994-007-5006-x.

Geoffrey I. Webb. Layered critical values: a powerful direct-adjustment approach to discovering significant patterns. *Mach. Learn.*, 71:307–323, 2008. doi: 10.1007/s10994-008-5046-x.

Yi-Leh Wu, Divyakant Agrawal, and Amr El Abbadi. Applying the golden rule of sampling for query estimation. *SIGMOD Rec.*, 30:449–460, May 2001. ISSN 0163-5808. doi: 10.1145/376284.375724.

Xin Yue Yang, Zhen Liu, and Yan Fu. MapReduce as a programming model for association rules algorithm on Hadoop. In *3rd Int. Conf. Inform. Sci. Interaction Sci. (ICIS), 2010*, pages 99–102, June 2010. doi: 10.1109/ICICIS.2010.5534718.

Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. In *Proc. 2nd USENIX Conf. Hot Topics in Cloud Computing*, HotCloud'10, pages 10–10, Berkeley, CA, USA, 2010. USENIX Association.

Mohammed J. Zaki, Srinivasan Parthasarathy, Wei Li, and Mitsunori Ogihara. Evaluation of sampling for data mining of association rules. In *Proc. 7th Int. Workshop on Research Issues in Data Eng.*, RIDE '97, pages 42–50. IEEE Computer Society, April 1997. doi: 10.1109/RIDE.1997.583696.

Mohammed Javeed Zaki. Parallel and distributed association mining: a survey. *IEEE Concurrency*, 7(4):14–25, October/December 1999. ISSN 1092-3063. doi: 10.1109/4434.806975.

Chengqi Zhang, Shichao Zhang, and Geoffrey I. Webb. Identifying approximate itemsets of interest in large databases. *Applied Intelligence*, 18:91–104, 2003. ISSN 0924-669X. doi: 10.1023/A:1020995206763.

Yanchang Zhao, Chengqi Zhang, and Shichao Zhang. Efficient frequent itemsets mining by sampling. In *Proc. 2006 Conf. Advances in Intelligent IT: Active Media Technology 2006*, pages 112–117, Amsterdam, The Netherlands, 2006. IOS Press. ISBN 1-58603-615-7.

Le Zhou, Zhiyong Zhong, Jin Chang, Junjie Li, J.Z. Huang, and Shengzhong Feng. Balanced parallel FP-Growth with MapReduce. In *IEEE Youth Conf. Inform. Comput. Telecom. (YC-ICT), 2010*, pages 243–246, November 2010. doi: 10.1109/YCICT.2010.5713090.