HEC MONTRÉAL

On a Wavelet-based strategy to estimate a distribution function on a distributed system

par

Akankhya Mohapatra

Jean-François Plante
HEC Montréal
Directeur de recherche

Sciences de la gestion
(Spécialisation M.Sc.)

Mémoire présenté en vue de l'obtention
du grade de maîtrise ès sciences
(M. Sc.)

May 2022

Résumé

Les systèmes distribués tels que Hadoop ont révolutioné la gestion des flux de données en offrant un cadre pour conserver et utiliser les données. Généralement, le goulot d'étranglement de tels système est la quantité d'information transmise d'un noeud à l'autre. Rassembler les données sur un seul noeud est géneralement impossible, ou trop coûteux. Cette thèse porte donc sur une stratégie de compression de l'information qui utilise les ondelettes.

Notre principal objectif consiste à estimer la fonction de répartition d'un échantillon univarié dispersé sur plusieurs noeuds. L'estimateur par ondelette que nous proposons est comprаré à un estimateur classique basé sur un sous-échantillonnage aléatoire des données. À l'aide d'une étude de Monte Carlo, nous présentons une analyse quantitative substantielle pour évaluer le potentiel de cette approche en variant différents paramètres. Les statistiques de Kolmogorov-Smirnoff (KS) et de Cramer-Von Mises (CVM) sont utilisées pour évaluer la performance.

Cette étude valide la pertinence des ondelette afin d'optimiser la communication entre les noeuds d'un système distribué. Les résultats de l'étude de Monte Carlo montrent des résultats prometteurs en ce sens.

Mots clés

Systèmes distribués, Hadoop, communication entre les noeuds, compression de données, ondelettes, fonction de répartition empirique, échantillonnage aléatoire, simulation de Monte Carlo

Abstract

Distributed systems such as Hadoop have revolutionized the management of incoming information streams, enabling a framework to store and process data. Generally, the bottleneck in such systems is the limitation in communication between the nodes. Querying all data at a single location is computationally expensive. Therefore, in this thesis, we adopt a wavelet-based strategy to compress the amount of information sent by nodes.

Our primary goal is to estimate the distribution function of a univariate sample stored across multiple nodes. The wavelet-based estimate that we propose is compared against a random sub-sampling benchmark. Using Monte Carlo simulation, we conduct an extensive quantitative analysis to study the effect of various parameters on the EDF estimate using wavelets. Kolmogorov-Smirnoff (KS) and Cramer Von Mises (CVM) statistics are used for evaluation measures.

This study establishes the relevance of wavelets in optimizing information communication between nodes in distributed systems. Based on Monte Carlo simulations, our study shows promising results in that respect.

Keywords

Distributed systems, Hadoop, communication between nodes, data compression, wavelets, Empirical Distribution Function, Random Sampling, Monte Carlo simulation

Contents

List of Tables

List of Figures

Acknowledgements

For a topic to be considered original on its own, a researcher must dare to choose a road less travelled by others. To demonstrate its potential, a researcher conducts extensive groundwork on the chosen topic and develops a unique methodology to accomplish the set goals. Being a novice in conducting research at such intensity, I set out on this journey quite excited and hopeful about learning it.

This work was written as part of my mandates for obtaining my master's degree at HEC Montreal, and it was undoubtedly one of the most magical - yet demanding - assignments I have undertaken so far. Throughout the work, I hope you will see my passion for portraying my learned skill and making this thesis a reality.

The people who have contributed to my thesis journey have left an indelible impression and cannot go unmentioned. First and foremost, I am very grateful to my thesis supervisor, professor Jean-François Plante, for accepting me under his guidance. This thesis would not have been possible without his mentorship and constant support. His continued motivation inspired me to excel in all aspects of my student life, which was necessary for the completion of my thesis. Special thanks to the people who supported me with funding for my studies at HEC Montreal and IVADO. It ensured my steady focus and commitment to my work.

Many thanks to the friends who kept me in high spirits during the pandemic and are like my family here. I would also like to thank my husband for his unwavering love and support and finally, my family back in India, who made this dream of mine of coming abroad to study attainable and supported me throughout.

Introduction

Despite the waning of the “Big Data” hype, the tools and challenges that once accompanied it have become mainstream. We live in an age when data rendering is of utmost significance. IoT sensors, social media, and healthcare applications increasingly generate massive amounts of data every day. Sending valuable insights learned and piloting business decisions from this data has never been more critical.

Data ubiquity presents challenges in storing large amounts of data. In recent years, data storage and processing systems for big data have become sufficiently stable to handle a large volume of rapidly changing data. The tools provided by distributed systems enable storage on multiple interconnected nodes. However, the rate at which data is communicated between the nodes is typically slow and must be minimized.

The idea of distributed systems has been around for more than two decades in computer science (e.g., Honavar et al., 1998, Mikler et al., 1998). The exploitation of distributed data through map-reduce, a framework designed specifically around distributed computing, contributed to the success of Google (Dean & Ghemawat, 2008). To that extent, Microsoft, Facebook, and Amazon web services now rely on hyper-scale computing of big data systems for storage and processing. Structured and unstructured data in big data systems with multiple cluster nodes depend more and more on distributed computation (Nataliya Shakhovska et al., 2019). The choice of processing in distributed systems is managed based on how much data is being input versus how urgent it is to process and consume data (Salma et al., 2017).

Several systems and algorithms have been developed for distributed data over the years (see Bhaduri et al., 2011 for a long list of contributions). In a survey of parallel approaches, Upadhyaya (2013) considers that the techniques for dealing with large size of data are mature. In this regard, however, there is a lack of algorithms for statistical inference suited to a distributed paradigm, and much work needs to be done (Jordan, 2013; Kleiner et al., 2014; Ma et al., 2015).

Caragea et al. (2004) consider models whose sufficient statistics can be represented as a sum. As long as each node maps the partial sum for sufficient statistics, the exact solution for the whole data set can be found by summing those partial sums together. Chu et al. (2007) take a closer look at the broader "Statistics Query Model," which consists of a group of models that can be computed using sum statistics. Despite their size, these families represent only a small fraction of the statistical models that may be needed on distributed data.

In their survey of statistical methods for distributed data, Wang et al. (2015) describe two strategies for distributed data: (1) sampling and (2) divide and conquer. Essentially, the sampling strategy indicates that only a subset of the data is used for inference. We will use this strategy as the baseline to exceed in our study. Divide and conquer is the second approach that has allowed linear models to advance significantly. A study by Lin & Xi (2011) use a first-order Taylor expansion to approximate estimating equations, while Chen & Xie (2014) study penalized regression.

In this paper, we focus on obtaining a good estimate of the distribution function of a single variable. A previous study of univariate distribution in a distributed context was conducted by Hu et al. (2007). The authors propose a gossip-based approach to estimate the kernel density by communicating regularly between pairs of nodes. By comparison, our method can be used in a map-reduce framework and does not require such heavy and frequent exchanges. We, however, assume that all nodes share a common sub-sample that they can leverage to encode their information.

Due to the limitation in bandwidth capacity in a distributed system, we assess a strategy to compress information sent across nodes. By using wavelets, data can be compressed by approximating the original function using a reduced number of coefficients.

Wavelet decomposition is comparable to Fourier series, the latter represented as an infinite sum of sinusoidal functions. Broadly speaking, both transforms can provide an informative mathematical and statistical understanding of various objects of interest such as functions, signals or images. Such representations can be obtained quickly through fast algorithms using an approximate number of coefficients (Ly Tran, 2006). Several surveys of wavelets and their applications indicate their significance in statistical inference (Anestis Antoniadis, 1997; Abramovich et al., 2000). However, with the inadequacy of the Fourier transform to deal with complexities in a function, wavelets received more attention (Amara Graps, 1995; Ly Tran, 2006), mainly because of their ability to capture local jumps in data. Wavelets use scalable modulated windows to capture details at different resolutions.

To determine the distribution of all data, an excellent place to start could be to examine the data that can be accessed from a single node within the same paradigm. However, accessing all of the data in such a distribution is not feasible. The use of inference methods based on statistical transforms is not new and has been considered by several authors before (Baringhaus et al., 1992). A large number of statistical methods for univariate data can be seen as a functionnal of the Empirical Distribution Function (EDF), notably goodness-of-fit tests, some non-parametric tests, as well as bootstrap. Therefore, we aim at estimating the EDF in our study, which can be used as a building block to develop such methods in other studies. As per Plante (2008), a good estimate of the EDF can even lead to an approximate likelihood method following the interpretation of the weighted likelihood.

We leverage a common sample shared with every node to determine the EDF of our univariate data. Our method uses a mapping based on the common sample, meaning every node can compute it with negligible communication costs. The EDF is represented in Haar wavelets decomposition (Haar, 1910), the simplest types of wavelets with a step-like representation. For retaining maximum information during compression of wavelet coefficients, Donoho and Johnstone $(1994, 1995)$ suggested the extraction of significant wavelet coefficients by thresholding. We consider hard and soft thresholding for our method. Wavelet transformation and thresholding of the univariate sample result in a reduced number of coefficients which is used for reconstructing the original function.

Referring to our earlier discussion on sampling strategy, a random hashing of our primary data, known as Simple random hashing (SRS), is considered as the baseline to the wavelet-based estimate. Kolmogorov-Smirnoff (KS) and Cramer-Von Mises (CVM) statistics are used as evaluation measures.

Overall, this study estimates a wavelet-based distribution function that exceeds the SRS benchmark. The compression technique using wavelets identifies coefficients with the maximum amount of information and consequently forces some coefficient values to zero. The communication of information between nodes with the reduced number of coefficients will result in occupying less space in node memory before transfer to another node for reconstructing the original function.

In chapter 1, we will present a more extensive literature review. Chapter 2 outlines the details of the methodology in our research. We include an extensive study of Monte Carlo simulations and a discussion of evaluation measures in Chapter 3. Finally, we conclude with prospective future directions.

Chapter 1

Literature Review

1.1 Big Data technologies

The drive towards digitization has evidently resulted in an increase in the rate of data generation embraced by different industries such as health care. Countless number of activities on different social media platforms and search engines also generate massive amounts of data every day. It is certain that data has become one of the most crucial commodities of recent times.

Such huge amounts of data produced needs to be processed and handled. Applications that rely more and more on real-time services demand a faster communication rate for data handling. In recent years, data storage and processing systems for big data have been sufficiently stable to handle a large volume of rapidly changing data. The tools provided by such distributed systems enable storage on multiple interconnected nodes.

Regardless, data-intensive applications are continuously growing the challenging concerns of the current distributed paradigm, especially for communicating information across the nodes (Choudhary et al., 2003). The work by Choudhary, et al. (2004) primarily focused on storage problems that many large distributed systems present on a regular basis. Their solution is to optimize principal I/O in a distributed environment by designing a distributed data archival system and accessing them locally. The architecture of their distributed file systems for accessing data makes an efficient use of a database for keeping the metadata of such file systems. It relies on using a database in a centralized network and accessing every file in a more local setting, with every file not available in the same network. It requires the usage of multiple storage resources heterogeneously distributed over networks.

Such instances resulted in much-needed exploitations of distributed data, eventually leading to the popularization of a framework explicitly designed around distributed computing, known as MapReduce (Dean & Ghemawat, 2004). The implementation of MapReduce revolved around the various approaches by which Google collected information by conducting a search analysis on website data for its search engine optimization. The primary motivation behind designing such a framework was understanding how to distribute data for storage and computation. MapReduce framework uses a map function, and a reduce function to associate a key/value pair for distinguishing each task while handling a large volume of data. Currently, Hadoop and Spark technologies actively use the MapReduce paradigm for distributed processing of massive amounts of data (Nataliya Shakhovska et al., 2019). These technologies are highly scalable and can sort through many clusters of nodes containing data in a comparably shorter period of time. To that extent, Microsoft, Facebook, and Amazon web services now rely on hyper-scale computing of Big Data tools for data warehousing and mining.

Collection of massive amounts of structured and unstructured data now depends more on the distributed paradigm for storage and processing (Nataliya Shakhovska et al., 2019). In a more practical application, a media streaming platform service such as Netflix streams videos using streaming data (unstructured) while at the same time, it may collect information on its platform usage by users in a structured manner. A domain-specific insfrastructure can be used for designing Big data technologies (Cigdem Avci Salma et al., 2017). Various features such as storage capacity and information extraction must be weighed based on the functional requirements of the domain of interest. Upadhyaya (2013) considers the Big Data tools dealing with large data considerably mature. The author discusses the use of MapReduce to deliver machine learning algorithms on GPUs.

Despite its usage in making domain-specific decisions, the buzz around solving statistical inferential problems using distributed paradigm is limited. It is essential to realize that statistical thinking could be a significant asset in driving such decisions forward. We, therefore, suggest discussing the inferential perspective in the “Big Data” context next.

1.2 Statistical Inference in a distributed infrastructure

Kleiner et al. (2014) proposed estimating statistics on a sample of data using a modified bootstrap approach, “Bag of Little bootstraps”. An accurate assessment of estimators on a small subset of data could benefit scaling to large datasets. Although the original bootstrapping approach assessed estimators in different settings, the resampling step causes data points to appear more than once, more so in a large dataset. Instead of applying an estimator (such as confidence interval) to assess the quality of estimate (such as population mean) to each small subset, their approach applies bootstrap to each small subset without having to rescale to the entire dataset.

Jordan et al. (2013) recommend that computational cost could be saved by uncovering solutions to statistical inferential problems in distributed infrastructures. They emphasized implementing the “divide-and-conquer” paradigm rather than depending on “computationally intensive” procedures to reach to the same solution. Additionally, they acknowledged Kleiner et al. (2012) recommendation to use small subsets of data because computing samples at independent processors (nodes) incurs a high computation cost. While “Bag of Little bootstraps” combines the good aspects of bootstrapping and subsampling, they highlight that there are instances where the method could be inconsistent in its performance. Their remark underlined the advantage of obtaining a good sample of data from the Empirical distribution function (EDF) as an approximation of the larger population, making the bootstrap method a functionnal of the EDF.

Jordan et al. (2013) also discuss a case of matrix factorization, where they use distributed computation in order to solve computationally cumbersome calculations of matrix dimensions. The discussion also reflects the importance of case-to-case model assumptions for obtaining a statistically meaningful approximation, which hinders developments in this area.

Another instance in support of using smaller subsets of data, works on a linear regression model with a fixed number of predictors to evaluate the statistical properties of algorithmic leveraging in a distributed paradigm while dealing with large datasets (Ma et al., 2015). Irrespective of standard sub-sampling approach, algorithmic leveraging utilizes sampling based on empirical leverage scores for making discoveries in the full data while using the smaller data portion as a "surrogate". They demonstrate the importance of statistical leverage in terms of bias and variance properties of the estimates in a parameter space much smaller than the sample size.

A paper by Caragea et al. (2004) considers an algorithm based on a learning strategy in distributed data. They based their method on "knowledge acquisition" systems from which learning can be possible, at one location with access to data before sending the learning to another required location or learnt from distributed nodes in cases with no access to data. Based on this general idea, it uses a minimal set of data to summarize parameter characteristics and generate a decision tree algorithm, which constitutes its sufficient statistics. A sufficient statistic is such that the exact likelihood of a model can be calculated from that statistic. It may be a single value or a vector of statistics depending on the model, but will often be of smaller dimension than the sample size. When sufficient statistics are expressed as a sum, every node may provide partial sums, that are recombined together to get an exact solution.

The study by Chu et al. (2007) examines the broader "Statistics Query Model," a set of models that can be computed using sum statistics in a distributed setting. Their study adopts a method for speeding up one algorithm at a time by utilizing additional cores in the computer system rather than focusing on algorithm optimization. Similar to the divide and conquer approach, the programming in this method relies on limited communication between the cores while using only a small fraction of the data points required for sufficient statistics calculation. It is only natural to calculate sum of data points in batches to exactly obtain the sufficient sums. Hence, all of the data is merged in a "summation" form, allowing speeding up processing in a distributed paradigm.

In their survey of statistical methods for distributed data, Wang et al. (2015) classified data processing essentially as sub-sampling (to draw subsets of data on which analysis can be made) and divide-and-conquer (parallel processing of each block before aggregation). Several other authors also considered the divide and conquer approach, such as Lin & Xi (2011), for estimating approximate equations using a first-order Taylor expansion. Chen & Xie (2014) used penalized regression on large sample size as a sub-sampling step for variable selection in a divide-and-conquer context. The divide-and-conquer approach resulted in a considerable speed-up determined empirically via asymptotic equivalence to an estimator based on a model using all data.

Statistical transforms have been used before for developing inference-based methods and are not new. Baringhaus et al. (1992) proposed goodness of fit test to estimate an unknown distribution's parameter characteristics based on a generating function. Using bootstrapping to estimate samples by Monte Carlo, they compared against alternative distributions for evaluating their proposed method.

A study of univariate distribution in a distributed context by Hu et al. (2007) proposed a "gossip-based" Kernel density estimation allowing frequent exchange between local nodes and estimating non-parametric statistics of unknown distribution from the same data. For an efficient estimation of distribution using nodes, the paper assumes an existing communication mechanism to "gossip" and exchange messages, making the local kernels available at every node to collect as much information on the distribution. The method depends on the communication between nodes to be adept at adequately estimating the distribution. By comparison, our method can be used in a map-reduce framework and does not require such heavy and frequent exchanges. We, however, assume that all nodes share a common sub-sample that they can leverage to encode their information.

The Empirical Distribution Function (EDF) can be essential in developing many statistical methods for univariate data, notably goodness-of-fit tests, non-parametric tests, and bootstrapping. Therefore, we are interested in estimating EDF in our study, which can be used to develop EDF-based approaches in other studies. The EDF is considered by

Plante (2008) as a good estimate to follow up on the interpretation of weighted likelihood to arrive at an approximate likelihood estimate.

One of the highlights from our above discussions is that communication between nodes supplements computation cost. Therefore, we aim to reduce the burden on every node by data compression in our strategy. We present the discussions surrounding the notion of data compression using wavelets below.

1.3 Wavelets

Various inferential applications of distributed systems still require much work to be done to reduce communication between nodes. Gathering all the data in a centralized location is generally neither desirable nor feasible for massive datasets because of bandwidth and storage constraints. In such systems, there is a need to assess a strategy for reducing the amount of information being exchanged between the nodes.

In our work, we use wavelets for compressing data by using a wavelet expansion, then approximating the original function with a reduced number of values. Such a transformation could also be obtained by an infinite sum of sinusoidal functions in an expansion of the Fourier series. Ly Tran (2006) describes a detailed study between these two types of transforms for approximating a function. The areas in which the Fourier series lacked became synonymous with growing applications of wavelets. During the decomposition and transformation of an object of interest (such as signals, functions, or images), the fast Fourier algorithm represents the object in a combination of sines and cosines. However, while dealing with complex data with sudden jumps, fast fourier algorithm does not implement well in the fixed window width which it uses for localization. Wavelets use the concept of modulating windows which are scalable in nature, making them more flexible in adapting to changes containing sharp jumps for function approximation.

Amara Graps (1995) describes the utility of fast algorithms to approximate a function using linear operations of Fourier transforms, and wavelet transforms. The flexibility of wavelets in dealing with local features in both time and frequency representations makes it more attractive for its usage in data compression, among its many other applications. Depending on the type of problem, a wavelet system is chosen to extract information as a "filtration" process, bringing out the "detail" information with smooth patterns.

Moreover, Wavelets have also been studied for their properties to help advance our understanding of multi-scale problems. Jawerth et al. (1992) provide a mathematical framework for constructing wavelet bases to apply multiresolution analysis. The generalization of one-dimensional wavelet transformation to a multi-dimensional case was made possible by Mallat (1989). The degrees of localization were based on the time-scale representation allowing the larger scale to capture extensive details and the smaller scale to capture minor details according to the requirement of an analytical problem.

Abramovich et al. (2000) considered wavelets-based methods for statistical problems such as in density estimation and nonparametric regression. Taking advantage of wavelet series expansion as a good basis, they applied a generalized Fourier series using a wavelet series to estimate fourier coefficients of a non-parametric function from a noisy data. For application of wavelets in density estimation, they discussed changes to the estimator form keeping the basic notion similar to that of a non-parametric regression. Wavelets allow functions in statistical problems to be approximated with fewer expansion terms based on the function and are therefore natural to use.

In statistical inferences, wavelets-based applications use orthonormal basis such as Haar (Haar, 1910) to compress objects of interest because of their compact support and step-like representation. Such studies lead to building statistical analysis algorithms in a closed form (Donoho and Johnstone, 1994). An excellent instance of the usage of wavelets in compressing data is by thresholding for recovering functions from noisy data. The portion of data containing noise was assumed to be "compressed" during recovery (Donoho and Johnstone, 1995).

An analytical study of communication between nodes in distributed systems is often complex, making it impossible to study their dynamics (Mikler et al., 1998). Therefore, there is a need to design a simulation specific to this area of research with the intention of increasing the efficiency of various statistical applications depending on the rate of communication between the nodes. In the next chapter, we detail our proposed method to estimate an EDF based on a compression strategy using wavelets.

Chapter 2

Methodology

The previous section discussed other studies about the different methods used to analyze communication in distributed systems. In this section, we present our proposed methods and expected results. We aim to study the distribution function of a finite univariate sample stored in a distributed system. We assume a small common sample on every node and use it to create a computationally inexpensive mapping structure. Wavelets are used to render a compressed estimate function. In the following chapter, we evaluate the resultant estimate using Kolmogorov Smirnoff and Cramer Von Mises statistics. We also use Monte Carlo simulation to evaluate their ability at reconstructing the original distribution function for various models.

2.1 Data and Notation

Distributed systems allow the information to be accessed from their nodes. Combined, nodes are repositories of vast amounts of data. As communication of information between these nodes is limited, the burden on the system to transfer data is enormous. It makes it difficult to send large amounts of data across nodes at a time. Therefore there is a need to minimize the amount of data communicated between nodes.

We consider a univariate sample of which we aim to determine the CDF for our study. The sample is a finite number of random observations and stored on different nodes in a distributed system. Our assumption includes a common sample that is replicated on every node in the system. That sample is leveraged for encoding the information before transferring data between nodes.

Keeping track of the minimum and maximum values in a distributed system is computationally inexpensive. So naturally, we keep these extrema from the common sample. For a chosen c, the common sample is comprised of $2^c - 1$ randomly selected data plus the minimum and maximum of the complete sample.

All nodes are given a common scale by creating a one-dimensional grid, consisting of non-overlapping grid cells whose count is related to a power of two. We leverage common sample points to obtain an equal number of grid cells. All the points of the common sample are part of the grid, which is further divided in intervals of equal length between any two values in the common sample.

To compress information, we depend on wavelet transformation of the mapped values based on the grid.

2.2 Mapping data to grid

Let us consider a distributed system having N interconnected nodes. Each node could be an individual commodity computer, for instance. The communication among the nodes in such systems is generally managed by Hadoop or Spark framework.

Let $i = 1, .., N$ be the index representing the N nodes in a distributed system. Let $X_{ij} \overset{iid}{\sim} F$ denote the j^{th} observation of n_i univariate data of interest in that node with distribution function F where X_{ij} is a set of independent observations. We have a common sample duplicated on every node, which is not comprised in our univariate sample. The common sample follows the same distribution F as the sample of interest, but also contains the min and the max in the complete sample. As a convention for mapping the data points, we express the common sample points in order statistics as X_{0j} with $j = 0, \ldots, 2^c$. A power of two is useful for our study given the mathematical decompositions that will be considered later. More explicitly,

$$X_{(00)} \leq X_{(01)} \leq \cdots \leq X_{(02^c)}$$

This indicates that X_{00} is the minimum of the whole dataset, and X_{02^c} is its maximum. Note since the data in the common sample does not appear in the exclusive samples X_{ij} for $i \geq 1$, the whole dataset has a total of $n = 2^c + 1 + \sum_{i=1}^{N} n_i$ data. This is inclusive of all data: the common sample containing the min and the max as well as the exclusive data from the N nodes.

Generally, the best way to study a sample following a particular distribution would be to consider the whole sample as if not distributed. Even if it were possible to find the true distribution by querying all data at one location, it would be highly cumbersome and computationally expensive.

The EDF estimates the law underlying the univariate data of interest. Using EDF, one can estimate the CDF for the points in a sample. Several classical methods, including goodness-of-fit tests, bootstrap, and some non-parametric methods, may be expressed as functionnals of the EDF. Our strategy provides alternative estimates for the CDF based on distributed data. A distributed system has limited bandwidth, so we aim to provide excellent performance while sharing limited information between the nodes.

The multiresolution property of the wavelet transform and its flexibility to deal with local features simultaneously in time and frequency provides suitable circumstance for expressing a function in a sparse representation. Multi Resolution Analysis (MRA) of functions carry information about the underlying distribution results in a series of approximations of functions analyzed at different resolutions. Fast transformations then encode the difference between these approximations using wavelets. Therefore, MRA and Fast wavelet transforms (FWT) allow representing our function of interest in the frequency domain, a space in which coefficients may be sparse. The sparse representation is not sensitive to local perturbations and therefore, we obtain a good approximation of the function.

MRA and FWT are defined on a bounded grid of values that are equally spaced.

Hence, we now define a mapping of the real axis to the $[0,1]$ interval using the ordered common sample that we previously defined. Each partition of the grid has common sample points separated by 2^c intervals. We also choose a maximum resolution of the mapping grid $G > c$ to define a finer grid of 2^G intervals by further dividing $[0,1]$ in intervals of equal lengths. A linear interpolation between the values of the common sample is used to map $[0,1]$ back to the original scale. Linear interpolation constructs new data points by using the location of two known points and rounding off the distance on the unit interval to be covered. So, we map X_{0j} to the value $j/2^c$, hence the minimum value of that sample is mapped to 0 and its maximum is mapped to 1 on the unit interval. In general, if $y \in [0,1]$, it will fall between mapped values of elements $a = \lfloor y2^c \rfloor$ and $b = \lceil y2^c \rceil$ of the common sample.

On the original scale, the value y will correspond to

$$x_{0a} + \left(y - \frac{a}{2^c}\right)(x_{0b} - x_{0a}).$$

To reiterate, the mapping depends on the common sample, so every node can compute it with negligible communication costs.

2.3 Estimates of the CDF

For the j^{th} of n_i univariate data of interest stored in node i, EDF may be defined as equal to the proportion of observations from the sample that are less than or equal up to that point. Assuming the data were in an ideal system equipped with unlimited computing resources, we could estimate F from the EDF of the entire dataset, namely,

$$\hat{F}(x) = \frac{1}{n} \sum_{(i,j)} \mathbb{1}_{(-\infty, X_{ij}]}(x).$$

where, $\mathbb{1}_{(-\infty, X_{ij}]}(x)$ is an indicator function equal to 1 when x is in (X_{ij}, ∞).

With limited resources, the estimation of F could be obtained using a random sample of data, say k of the n points and compute $\hat{F}^{(k)}$, the EDF on that smaller sample. Our strategy estimates the CDF from limited information in a distributed system. We set an information budget k to limit the number of resources allocated. Therefore, we consider the corresponding $\hat{F}^{(k)}$ as a benchmark that we aim to outperform using our method of estimation. The sampling approach includes having a subset of smaller number of observations from the original sample with each X_{ij} having an equal chance of getting selected at random.

2.3.1 EDF based on our method of compression

Let $\hat{F}_i$ denote the EDF of the exclusive data of node i on its original scale. The exclusive data does not include the common sample copy stored at every node. However, in order to determine the EDF of the exclusive dataset on the unit scale $[0,1]$, it must leverage the copy of the common sample stored on all nodes.

Using our previous knowledge of the mapped values of elements on the unit scale, the exclusive EDF becomes,

$$\hat{H}_i(y) = \hat{F}_i(x_{0a}) + \left(y - \frac{a}{2^c}\right)\left\{\hat{F}_i(x_{0b}) - \hat{F}_i(x_{0a})\right\}$$

where, $a = \lfloor y2^c \rfloor$ and $b = \lceil y2^c \rceil$. The EDF of the common sample results in a right continuous step function which is the same at every node. The EDF of the common sample is,

$$S(y) = \min\left\{\max\left(0, \lfloor y2^c \rfloor / 2^c\right), 1\right\}.$$

In order to determine the specificities of the EDF of node i, we compute the empirical error between the exclusive data EDF estimate $\hat{H}_i$ and the EDF based on the common sample S at a node i. Our strategy comprises of making nodes share their difference with respect to the common sample through the functions $\hat{E}_i = \hat{H}_i - S$.

The difference between the two EDF carries information about the law underlying the distribution function of the sample of data at node i. We aim to compress that EDF difference using wavelets. We may note here that $\hat{E}_i(0)$ may vary when it takes the value of the first step while $\hat{E}_i(1) = 0$ because both $\hat{H}_i(1) = 1$ and $S(1) = 1$. As we discuss the details of wavelets in the next section, we define $\tilde{E}_i$ as the compressed version of $\hat{E}_i$ in the sense that it is obtained using a reduced number of non zero values indicative that our method adapts a subdivision scheme giving a way for reconstruction of the global estimate of F.

We reconstruct a global estimate for F on the basis of the compressed EDF $\tilde{E}_i$ by allowing the collection of information shared by the N nodes as well as the common sample. This proportion of information at node i and across all N nodes is used to rebuild the function $\tilde{E}_i$ and we then obtain,

$$\tilde{H}^*(y) = S(y) + \sum_{i=1}^{N} \frac{n_i}{n} \tilde{E}_i(y),$$

where, $\tilde{H}^*(y)$ is the wavelet approximated CDF of the whole sample on the unit scale. By waiting until the end to divide by n and add $S(y)$, this combination is suitable for a reduce step in a map-reduce context.

The reconstructed distribution function may lose its monotonicity because of the compression. Hence we make an effort at fixing it to yield a linear piecewise step function defining the reconstructed $\tilde{H}^*(y)$ as a representation of the original function after compression. To preserve the order of the function similar to that of our original function, we apply isotonic regression (see e.g. Barlow et al., 1972) to $\tilde{H}^*$ which yields an increasing estimate $\tilde{H}$ similar to the original function $\hat{H}$. Reverting to the original scale, the corresponding approximate CDF $\tilde{F}(x)$ may be obtained through the usual mapping we discussed earlier. Although Isotonic regression is computationally demanding, it is applied only on the last node and hence does not affect the communication between the nodes which is the real bottleneck in our situation.

2.3.2 EDF based on SRS

As a benchmark, we consider an EDF based on a sample of k points selected from the union of all the available data. That EDF is noted $\hat{F}^{(k)}$ and is evaluated on the same grid points as the other functions it is compared to. To compute $\hat{F}^{(k)}$, k data points need to be transferred to a central node. It is therefore a fair comparison point for a wavelet-based estimate with k non-zero coefficients. In a way, k is our information budget, and we compare newly proposed methods to an EDF made from a commensurate subsample.

2.3.3 EDF based on true distribution

In order to set a common scale for evaluation, we are in need of a reference distribution function for fair comparison and analysis. Since both the wavelet based EDF and the SRS based EDF are based on grid mappings, we consider the true EDF of complete dataset based on the developed grid as well and denote it as the true distribution function $F(x)$ for our study.

2.4 Wavelets

Wavelets are families of functions that may be used as a basis for decomposing functions on a bounded interval. The multiresolution property of wavelets treats local features yielding a sparser representation of the local variations in $\tilde{E}_i$ with few significant values representing its underlying distribution. Our function of interest, $\hat{E}_i$ is defined on $[0,1]$, a bounded interval ("wavelet probing" by Anderson et al., 1994). In addition, it is a step function defined on 2^G intervals of equal length spanning the unit scale.

Haar Wavelets, the simplest of all commonly used wavelets, yield precisely that type of step function, dividing the total span with two step functions into ranges of 0 to $\frac{1}{2}$ and $\frac{1}{2}$ to 1. The Haar bases have two mutually orthonormal parent functions i.e. the mother wavelet $\psi(y) = \mathbb{1}_{[0,1/2)}(y) - \mathbb{1}_{[1/2,1)}(y)$ and the scaling function $\phi(y) = \mathbb{1}_{[0,1)}(y)$ referred to as the father wavelet.

The size of each coefficient associated with the wavelet function can vary considerably between different levels of resolution depending on the information detail to be retrieved. Hence, in a Multi-resolution analysis (MRA), Haar bases will provide coarser to finer details by linear combinations of scaling and translating these two functions, namely,

$$\phi_{s,\ell}(y) = 2^{s/2}\phi(2^s y - \ell)$$

$$\psi_{s,\ell}(y) = 2^{s/2}\psi(2^s y - \ell)$$

where s and ℓ are scale and position parameters respectively which are used to vary the resolution level. We need the MRA to compress a function on the bounded interval $[0,1]$. There is no need to consider resolutions below $s = 0$.

For the function $\hat{E}_i$ defined on 2^G steps, the maximum resolution will be $s = G$. Hence, $s = 0,\cdots,G$ and $\ell = 0,\cdots,2^s - 1$ for our purpose. Intuitively, from the above equations, a unit increase in s will have an effect on $\psi_{s,\ell}$ into half the width doubling its estimate in frequency terms, i.e. as s would increase, the functions may become narrower. A unit increase in ℓ will shift the location of both $\phi_{s,\ell}$ and $\psi_{s,\ell}$ by an amount proportional to their width 2^{-s} (Abramovich et al ., 2000).

Let $\hat{E}(y)$ represent a generic $\hat{E}_i$ for a single node i. At resolution G, we can express $E(y)$ as

$$\hat{E}(y) = \sum_{\ell=0}^{2^G-1} e_{G,\ell}\phi_{G,\ell}(y)$$

where, $e_{G,\ell} = 2^{-G/2}\hat{E}\{(\ell-1)/2^G\}$. Using the recursive properties that $\phi_{s,\ell} = 2^{-1/2}\phi_{s+1,2\ell} + 2^{-1/2}\phi_{s+1,2\ell+1}$ and $\psi_{s,\ell} = 2^{-1/2}\phi_{s+1,2\ell} - 2^{-1/2}\phi_{s+1,2\ell+1}$, the function $\hat{E}(y)$ may also be represented in frequency space as

$$\hat{E}(y) = e_{0,0}\phi_{0,0}(y) + \sum_{s=0}^{G}\sum_{\ell=0}^{2^s-1} d_{s,\ell}\psi_{s,\ell}(y)$$

where the coefficients $e_{0,0}$ and $d_{s,\ell}$ can be found in linear time using a FWT. The equation above represents an expansion series of $\hat{E}(y)$ as a wavelet series of successive approximations on the interval $[0,1]$. In this new frequency space, those coefficients may be sparse. The first approximation term is achieved by the scaling terms $e_{0,0}\phi_{0,0}$. It is possible to approximate local features that cannot be adequately captured by the scaling terms in frequency space and correspondingly fine detail by sequences of wavelet terms $d_{s,\ell}\psi_{s,\ell}$.

Compression may hence be achieved by nullifying some coefficients associated with the wavelet function to zero with the remainder wavelet coefficients containing significant bits of information. The communication of information between the nodes with the reduced number of coefficients may result in occupying less space in node memory before getting transferred to another node for the reconstruction of the original function as $\tilde{F}(x)$.

2.5 Compression strategy

The approximation of $\hat{E}_i$ at a resolution 2^{s+1} comprises all information to compute the same function at a smaller resolution 2^s given that the function at 2^s is at a coarser scale than at 2^{s+1}. This indicates that the function at 2^{s+1} can be represented as a sum of the function at 2^s scale (Sheng Y., 2000). However, our method is used to filter out only the significant bits of information to reconstruct the function as $\tilde{E}_i$ using compression strategy with wavelets.

To develop an efficient method, information about the functions $\hat{E}_i$ has to be compressed before it is sent to other nodes. Throughout this section, let $\hat{E}(y)$ represent a generic $\hat{E}_i$. It is defined over 2^G values, and can hence be represented as the vector of discrete values $(e_1, \cdots, e_{2^G})^T$. The fast wavelet transform moves these values to a different space, the frequency space. Initially, the wavelet transform counts as many parameters, but it is likely to be sparse, with multiple values that are close to zero.

2.5.1 Wavelet Transform

For the compression strategy, we consider a Discrete Wavelet transform (DWT) to consist of 1) the functions $\phi_{s,\ell}$ and, 2) a method to determine the coefficients $e_{s,\ell}$ to be chosen from the wavelet transform. Using the generic expression of $\hat{E}(y)$ described in the *Wavelets* section, we will define the wavelet decomposition of function $\hat{E}(y)$ such that our proposed method calculates the compressed coefficients $\tilde{e}_{s,\ell}$ and the compressed function will take the form

$$\tilde{E}(y) = \sum_{\ell=0}^{2^s-1} \tilde{e}_{s,\ell}\phi_{s,\ell}(y)$$

We obtain detailed coefficients directly from the higher frequencies and approximation coefficients from the lower frequencies which are subsequent down-sampled versions of the original function $\hat{E}$ as a result of the wavelet transform. The level of information the decomposed coefficients will contain are a clear consequence of this step in DWT.

Followed by wavelet decomposition, compression occurs during wavelet thresholding process. Compressed coefficients are used for reconstruction of the original function as $\tilde{F}$. The process of reconstruction has been explained in the *Estimates of the CDF* section earlier.

For determining $\tilde{e}_{s,\ell}$ with m non-zero values, we choose the m largest values from the 2^G coefficients in $e_{s,\ell}\phi_{s,\ell}$. We set a bound of choosing $m << 2^G$ distinct values in the frequency space which allows us to select coefficients of $\tilde{e}_{s,\ell}$ with a smaller number of significant bits.

We use both hard and soft thresholding types and possible dependency on decomposition level during the wavelet thresholding to extract m coefficients. Considering a threshold λ and a generic representation of wavelet coefficients from thresholding as e_{thresh} , hard thresholding follows a rule for keeping absolute values of wavelet coefficients as $e_{thresh}\mathbb{1}(|e_{thresh}| > \lambda)$ while soft thresholding follows the rule of shrinking the wavelet coefficients it keeps as $sgn(e_{thresh})\max(0, |e_{thresh}| - \lambda)$ (Donoho and Johnstone , 1994, 1995). We use k to limit the number of compressed coefficients that can be chosen during the process.

We also consider “wavelet level dependency” during wavelet transform and thresholding. Implicitly $\psi_{s,\ell}$ is scaled by a power of square root of two. In combination with the wavelet thresholding, the threshold λ is now multiplied by a factor of $2^{-s/2}$ to account for this. This results in a wavelet “level dependent” thresholding or adaptive thresholding, adjusting the scale to further accommodate for a more precise selection of m number of wavelet coefficients.

The value of m coefficients during wavelet thresholding ultimately forces some ties to zero, which may lead to reduced space in the node memory. This means that the transfer of information between nodes involves m coefficients as well their position in the wavelet decomposition, an information that may be efficiently coded with a few bits and whose cost will be considered negligible.

2.5.2 Inverse Wavelet transform

Using the same convention as before in the function $E(y)$ emplying the recursive properties of $\phi_{s,\ell}$ and $\psi_{s,\ell}$ (in the *Wavelets* section), we perform the 1-D inverse DWT of $\{\tilde{e}_{s,\ell}, \tilde{d}_{s,\ell}\}$ for reconstructing the function of $\tilde{E}$ using the compressed coefficients, yielding,

$$\tilde{E}(y) = \tilde{e}_{0,0}\phi_{0,0}(y) + \sum_{s=0}^{G}\sum_{\ell=0}^{2^s-1} \tilde{d}_{s,\ell}\psi_{s,\ell}(y)$$

where $\tilde{e}_{s,\ell}$ and $\tilde{d}_{s,\ell}$ are compressed coefficients of $\phi_{s,\ell}$ and $\psi_{s,\ell}$.

Intuitively, every level 2^{s+1} recursively generates the vector at the previous coarser level 2^s until finally $\tilde{E}$ is obtained. We may refer back to the *Estimates of the CDF section* for the reconstruction of the wavelet approximated CDF of the whole sample $\tilde{H}^*$ on the unit scale and finally obtain the global estimate $\tilde{F}$.

Chapter 3

Simulation Results

In this chapter, we want to assess the effectiveness of the proposed method. We use Kolmogorov Smirnoff (KS) and Cramer Von Mises (CVM) statistics to measure the quality of the EDF estimate based on wavelets. We describe Monte Carlo simulation for performing our experiments. We also discuss the results of the simulations conducted.

3.1 Evaluation Measures

In order to estimate the distribution function, the wavelet based compression depends on simulating the samples using a random generator function. In an ideal system, $\hat{E}_i$ and compressed estimate $\tilde{E}_i$ would be considered the same and the error between them would be zero. However, by achieving higher compression levels, we observe differences in the compressed estimate $\tilde{E}_i$ from $\hat{E}_i$.

We evaluate the effectiveness of the EDF estimate based on our method by comparing $\tilde{F}(x)$ against a random sample of data, k of n data points, as our benchmark $\hat{F}^{(k)}(x)$ for a given information budget k. We use KS and CVM statistics to evaluate the difference between the two measures in terms of distance. We earlier discussed the estimation of true distribution of the univariate sample of interest using the Grid map. The distance between the two measures is quantified by the true EDF estimate F in terms of ratio.

The evaluation procedure entails calculating the distance between $\hat{F}^{(k)}(x)$ and $F(x)$

in the first iteration. In the second iteration, we calculate the distance between $\tilde{F}(x)$ and $F(x)$. We then determine the ratio between the two iterations using KS and CVM statistics, namely

KS statistics uses the largest absolute difference between the EDF measures defined as

$$d^{KS} = \frac{\sup_x |\hat{F}^{(k)}(x) - F(x)|}{\sup_x |\tilde{F}(x) - F(x)|}$$

CVM statistics uses the mean difference between the EDFs in comparison with respect to their iterated differences, defined as

$$d^{CVM} = \frac{\int_x [\Big(\hat{F}^{(k)}(x) - F(x)\Big)]^2 dF^(x)}{\int_x [\big(\tilde{F}(x) - F(x)\big)]^2 dF^(x)}$$

Both d^{KS} and d^{CVM} are determined numerically. In the case of d^{CVM}, a numerical integration is performed using the grid that we defined earlier in Chapter 3.

3.2 Monte Carlo Simulation

We want to evaluate the performance of our method using wavelets to estimate EDF realistically by running several simulations. By using Monte Carlo (MC) simulation methods, we repeatedly draw samples through several runs of simulation and evaluate the EDF under each Model. Every run will produce an EDF based on the random set of samples in that Model which is evaluated using KS and CVM statistics. The samples are generated using random hashing and random hashing with ordered hashing, i.e., where the smallest values are attributed to a node, the next smallest to another node and so on.

While constructing the simulation, we also generate the univariate samples based on various distributions. Our method will compress information sent across nodes using wavelets which depends on some parameters we vary during simulations to observe its effect in estimating EDF. Parameter variation is performed according to various distributions we want to study.

We use a total of 100 repetitions to produce and evaluate the wavelet-determined EDF estimate. We set the seed of the generator to ensure reproducibility. The simulation is run on my computer server memory which may take up to few hours to run. The simulation results are saved to an *Rdata* file and every subsequent run loads the data from the file for generating tables and plots.

3.3 Distributions

For our study, we use the following four cases of distributions for evaluating the effectiveness of our proposed method:

Table 3.1: Different Parameters of all distributions considered for our study

Normal	Cauchy	Gamma	Mixture of Normals
$Mean$	$Location$	$Rate = \frac{1}{Scale}$	$Means$
$Standard\ Deviation$	$Scale$	$Shape$	$Standard\ Deviations$
			$Mixture\ Proportions$

Depending on the requirement for our study, we use Normal distribution as a standard go-to model, Cauchy distribution for studying robustness of model with extreme values, Gamma distribution for positive values and Mixture of Normals for introducing variability with multimodal distributions in our study. We now specify the parameters of each distribution type. We consider standard values with $mean\ = 0$ and $standard\ deviation\ = 1$ for normal distribution.

For Cauchy distribution, we have two parameters: $Location$ and $Scale$. Both parameters have been considered standard values with 0 and 1 respectively.

For Gamma distribution, the two parameters are: $shape$ parameter and $rate$ or $scale$ parameter, where $rate\ = \frac{1}{scale}$ and $shape \geq 1$. In our study, we consider $rate\ = 1$ and $shape\ = 2$.

Mixture of Normals have 4 equiprobable components with each component having $(0,2,4 \text{ and } 8)$ means and 0 standard deviations.

3.4 Simulation of data

The simulation is done using *R* with the *Rstudio* 3.6.1 interface. For our simulation, we adapt two strategies of distributing our data across nodes. The recipe to send a given data to a given node is called hashing. We consider:

i. random hashing yielding an exclusive sample with $n_1 = n_2 = \cdots = n_i$ at every node i , where $i = 1,2,\cdots,N$

ii. random hashing with ordered hashing yielding an exclusive sample where the values of node i_1 are less than those of node i_2 if $i_1 < i_2$, where $i_1, i_2 \in i$ and $i = 1,2,\cdots,N$.

Random hashing ensures the best way to generate observation $X_{ij} \sim F$ for node i with each selection of observation generated randomly. This type of simulation preserves the order in which they are generated. In the second type of data simulation, we use ordered hashing to place the observations X_{ij} generated in every node i in an increasing manner.

In the main simulation study, we only consider balanced nodes i.e. every node will have the same sample size. The wavelet transform is performed on a discrete grid of size related to a power of two having 2^G grid cells.

For each grid cell, there are attribute independent parameters (calculations for which are independent of the grid cell) and attribute dependent parameters (calculations of which depend on the grid cell).

i. Attribute independent parameter: min, max

ii. Attribute dependent parameters: estimates of EDF

min – minimum point of the Grid; max – maximum point of the Grid

Value of Min and Max are calculated based on the simulated univariate sample and the common sample according to different distributions considered.

All estimates of EDF based on compression strategy using wavelets, SRS and true distribution are based on the Grid.

3.5 Simulation flow

We aim to study the distribution of information exchanged between nodes using EDF. We use wavelet transformation and thresholding to compare against an SRS benchmark. However, the compression method using wavelets will generate different compressed functions $\tilde{E}_i(y)$ depending on some parameters which we allow to vary.

Using MC simulation, we study the effect of varying the four sets of parameters below:

$i)$ the grid size G,

$ii)$ the grid size G in combination with the chosen constant c for the assumed common sample,

$iii)$ the total number of nodes N, and

$iv)$ the total number of nodes N in combination with the sample size at each node n_i, where $i = 1, .., N$

on the estimation of wavelet-based EDF $\tilde{F}(x)$. These variations are performed on both individual level and in combination to assess the functionality of our proposed method. We will compare each Model on the basis of their evaluation measure using KS and CVM statistics.

Wavelet thresholding is initially varied between hard and soft thresholding types along with dependency on decomposition level before proceeding with other simulations. The benchmark SRS is varied related to the power of two according to the grid size 2^G. We try different combinations of each parameter values and vary it until it meets the available memory space of the computer on which the experiments are run.

The study is also conducted under the effect of two types of data simulation and four cases of distributions described above. We begin with Normal distribution and Cauchy distributions varying all sets of parameters and observing the outcomes of each simulation. MC simulation is carried out through out all experiments to have a common scale of observation of function characteristics. Consider the case where similar results are observed for a set of parameter variations in two distributions; then, we extend the interpretations of the obtained results to the other two distributions (Gamma & Mixture of

Normals) without rerunning.

We place a stronger importance on the reconstruction of the EDF based on Wavelets and their evaluation by using the same structure of constructing the simulation Models for our study. All Models share the same number of fixed and variable parameters. The range of values for variation for every parameter is fixed for the first two sets of simulation studies comprising of Normal distribution and Cauchy distribution type. It is amended accordingly for the other two distributions based on the outcomes of the earlier simulations. We now present our scenarios below.

3.6 Default Scenario

Our proposed method depends on multiple parameters. We define a default scenario consisting of fixed parameter values. Our hope from experimenting with a default scenario is to understand the efficiency of our method in reconstructing the compressed coefficients approximating the original function. Such an experiment will not weigh heavy on computation and results from the implementation would be quick and useful for planning our simulations using MC method. This scenario will also serve the purpose of a reference while working with complex scenarios for fair comparison in terms of performance.

The univariate sample of interest distribution is generated by assuming a normal distribution such that it yields X_{ij} independent random variables $\sim \mathcal{N}(0,1)$, where for $i = 1,..,N$ number of interconnected nodes, the node having n_i univariate data that we are interested in is represented by the j^{th} variable. The complete data excluding the common sample contains the data of one node forcing equal sample sizes of X_{ij} with j^{th} denoting $n_i = 500$ data points on every node i of the $N = 20$ nodes in total.

The common sample consists of the 15 sample points shared by every node i, for a constant value of c chosen to be 4. We choose the grid map size $G = 8$ which yields a grid resolution of size $2^G = 256$ including the minimum point X_{00} and maximum point X_{02^c} for the whole dataset. Further, linear interpolation yields 16 partitions in each grid cell.

For the number of coefficients to be transferred between the nodes, we set $k = 32$

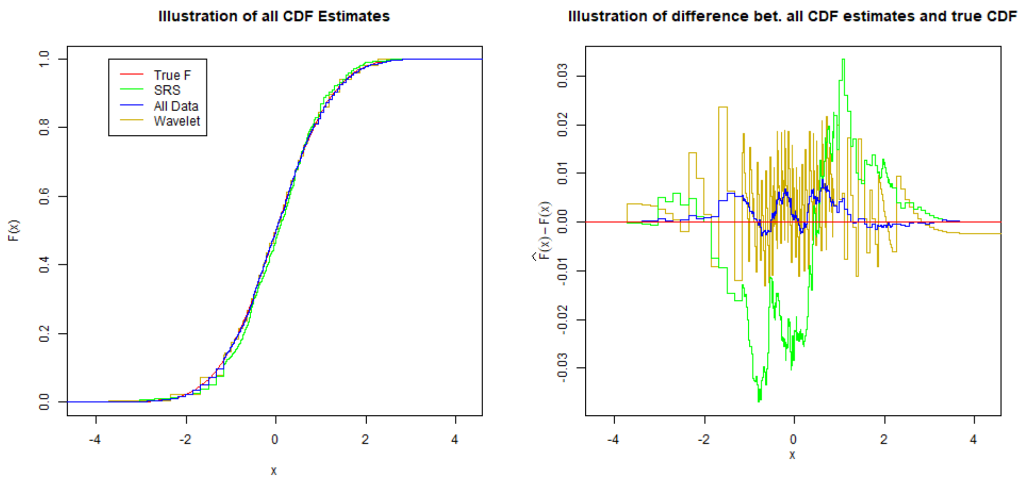

Figure 3.1: Illustration of CDF estimates and their differences in default scenario

generating 640 data points. From the $N = 20$ nodes, the total sample becomes 10,000 points. Along with the minimum and maximum points and the common sample of 15 data points, we will have a total of 10017 data points. The SRS yields different number of data points at every node, that is, they are not all accumulated at at a single node, rather randomly spread across all nodes in different numbers.

Using hard thresholding without level dependency during wavelet transformation, the resulting coefficients are compressed and are used to reconstuct the EDF $\hat{F}$ approximating the original distribution function F. Monotonicity of the estimated EDF $\hat{F}(x)$ is taken care of using the $isoreg()$ function in R.

We present two plots of different estimates of CDF based on wavelets $\tilde{F}$, on SRS $\hat{F}^k$ and the complete data $F_{undistributed}$ with respect to the true distribution CDF F in fig 3.1.

The left panel in fig 3.1 shows the step function of various CDF estimates. The plot in *red* shows the true CDF F. It is only natural to find that as the original function is monotonically increasing in nature, the EDF based on wavelet $\tilde{F}$ and SRS $\hat{F}^k$ will also be monotonically increasing. The points that are different from the simulated data points are represented as “steps” towards the edge of the curve on both ends in the CDF estimates based on SRS $\hat{F}^k$ (in *green*) and wavelets $\tilde{F}$ (in *gold*) .

Table 3.2: Illustration of CDF estimates using random hashing with Normal distribution having $c = 4$, $G = 8$, $N = 20$, $n_i = 500$, $k = 32$ in default scenario

	KS	CVM
SRS	0.0369685	0.0000016
Wavelet	0.0235495	0.0000004
Wavelet Ratio	1.5698187	4.0636106

The right panel of fig 3.1 shows the CDF estimates based on the wavelets $\tilde{F}$ and SRS $\hat{F}^k$ and of their difference in comparison to the true CDF F. The wavelet based CDF estimate $\tilde{F}$ (in *gold*) and SRS based CDF estimate $\hat{F}^k$ (in *green*) are more visibly distorted than the CDF of the undistributed complete data $F_{undistributed}$ in *blue*. The plot of wavelet EDF $\tilde{F}$ is more concentrated in the middle than in the edges which could be indicative of the function approximation using reduced number of coefficients during wavelet transform and thresholding. We also observe the CDF estimate based on SRS having the peaks much higher from the line of reference $\hat{F}^k$ (in *red*), indicating the approximation of the piece-wise function based on random selection of coefficients in SRS.

To evaluate our study, KS and CVM statistics are used to measure the distance between the CDF estimates based on wavelets and on SRS obtained with the same communication budget. Naturally, if the ratio between them is greater than one, the wavelet based estimate is closer to the true CDF. As the values in KS and CVM ratios suggest in table 3.2, the model without the MC simulation yields a good estimate of CDF based on the compression method using wavelets.

The parameter values in this scenario are used as a reference for the range of values to be varied in the subsequent scenarios for simulation. All further experiments are based on variations from the parameters in this scenario with the intention of improving upon this result of evaluation based on KS and CVM ratio statistics. This will qualify to witness the extent to which our proposed method remains effective.

Before moving on to performing experiments using MC method, we display a list of parameters that we consider to vary below.

Table 3.3: An overview of all parameters considered for variation in our experiments

Exp.	Dist.	Data sim. **	k	Thresh	Lev dep.	G	c	N	n_i
1	Normal	RH	×						
2	Normal	RH		×	×				
3	Normal	RH				×			
4	Normal	RH				×	×		
5	Normal	OH				×			
6	Normal	OH				×	×		
7	Normal	RH						×	
8	Normal	RH						×	×
9	Normal	OH						×	
10	Normal	OH						×	×
11	Cauchy	RH				×			
12	Cauchy	RH				×	×		
13	Cauchy	OH				×			
14	Cauchy	OH				×	×		
15	Cauchy	RH						×	
16	Cauchy	RH						×	×
17	Cauchy	OH						×	
18	Cauchy	OH						×	×
19	Gamma	RH				×	×		
20	Gamma	OH				×	×		
21	Gamma	RH						×	×
22	Gamma	OH						×	×
23	Mixture	RH				×	×		

Exp.	Dist.	Data sim. **	k	Thresh	Lev dep.	G	c	N	n_i
24	Mixture	OH				$\times$	$\times$		
25	Mixture	RH						$\times$	$\times$
26	Mixture	OH						$\times$	$\times$

** RH = Random hashing, OH = Ordered hashing

where, Exp = Experiments or Scenarios, Dist = Distributions, Data sim = method of univariate sample simulation, k = information budget, thresh = thresholding type, lev dep = wavelet decomposition level dependency, G = Grid resolution, c = common sample size, N = Number of nodes, n_i = sample size in each node

We now perform experiments for each scenario and interpret the results in the next section.

3.7 Normal Distribution

3.7.1 Scenario 1 - effect of varying SRS value with Random hashing

For scenario 1, we consider varying SRS budget k with normal distribution using random hashing for generating a univariate data of interest. We focus on sending as much information as possible with minimal burden on the system.

We accordingly set the value of common sample size c, the grid resolution G, the number of nodes N and their sample sizes n_i to the values used in the default scenario. We use *hard* thresholding without dependency on the wavelet decomposition level $lev\ dep$.

For $k \in \{32, 64, 128\}$, the table 3.4 shows the effect of varying SRS on the KS and CVM ratios:

We observe that the KS and CVM performance ratios seem to decrease as k increases. However, as all the ratios are greater than one, we can say that the scenario 1 is yielding

Table 3.4: KS and CVM ratios for varying SRS using random hashing with normal distribution having $G = 8$, $c = 4$, $N = 20$ and $n_i = 500$ in Scenario 1

k	KS performance ratio	CVM performance ratio
32	1.888	4.988
64	2.139	4.833
128	1.499	2.031

Table 3.5: KS ratio for varying threshold and decomposition level using random hashing with normal distribution having $G = 8$, $c = 4$, $N = 20$, $n_i = 500$, $k = 32$ in Scenario 2

$lev\ dep$	$thresh$ = hard	$thresh$ = soft
TRUE	1.572	0.614
FALSE	1.888	0.951

Table 3.6: CVM ratio for varying threshold and decomposition level using random hashing with normal distribution having $G = 8$, $c = 4$, $N = 20$, $n_i = 500$ and k = 32 in Scenario 2

$lev\ dep$	$thresh$ = hard	$thresh$ = soft
TRUE	5.179	0.718
FALSE	4.988	2.330

better results with a smaller budget k. We set $k = 32$ for all subsequent simulations.

For our next simulation, we consider varying the thresholding type and wavelet decomposition level dependency.

3.7.2 Scenario 2 - effect of varying thresholding type and wavelet decomposition level with random hashing

In this scenario, we want to observe the effect of replacing hard thresholding by soft thresholding and the dependency on the wavelet decomposition level on the KS and CVM ratios. Considering k = 32, we keep all the other parameter values the same as $G = 8$, $c = 4$, $N = 20$ and $n_i = 500$. We use random hashing for generating a univariate sample.

In tables 3.5 and 3.6, we can observe the effect of replacing the thresholding type $thresh = hard$ to $soft$ and wavelet decomposition level dependency $levdep = false$ to $true$.

Table 3.7: KS and CVM ratios for varying G using random hashing with normal distribution having $c = 4$, $N = 20$, $n_i = 500$, $k = 32$ in Scenario 3

G	KS performance ratio	CVM performance ratio
8	1.572	5.179
10	1.510	4.413
12	1.389	4.071
14	1.388	3.917
16	1.382	3.894
18	1.381	3.884

All the parameter variations yielded KS and CVM ratios close to one except for soft thresholding with level dependency. The simulation run on only hard thresholding with dependency on the wavelet decomposition level yielded KS and CVM performance ratios comparable to the results of hard thresholding without the level dependency. As the performance of hard thresholding is better than soft thresholding both with and without level dependency, henceforth for further scenario simulations, we will be considering *hard* thresholding with dependency on the wavelet decomposition level $lev\ dep = true$ only.

3.7.3 Scenario 3 - effect of varying G using random hashing

In scenario 3, we consider to observe the effect of varying a single parameter, the grid size G on the KS and CVM ratios using random hashing for generating a univariate data of interest. The effect of varying Grid resolution G on the resulting performance ratio would indicate whether G has any effect on the outcome ratios.

We consider *hard* thresholding type and wavelet decomposition level dependency $lev\ dep$ as $true$, and vary G between the values $\{8, 10, 12, 14, 16, 18\}$. All other parameter values are kept the same with $c = 4$, $N = 20$, $n_i = 500$ and $k = 32$.

We may refer to the table 3.7 for studying the results of the variation on the KS and CVM ratios.

Looking at the results, we observe all the KS and CVM performance ratios are greater than one, indicating the effectiveness of our method in this scenario. However, the ratios decrease as G increases.

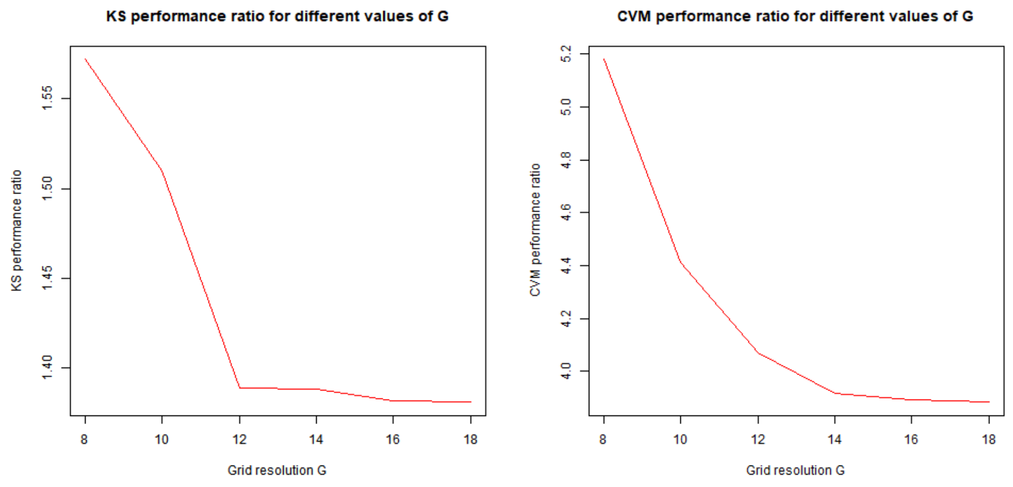

Figure 3.2: Illustration of the effect of varying G on the KS and CVM ratios using random hashing with Normal distribution having $c = 4$, $N = 20$, $n_i = 500$ and $k = 32$ in scenario 3

Table 3.8: KS ratio for varying G and c using random hashing with Normal distribution having $N = 20$, $n_i = 500$, $k = 32$ in Scenario 4

c	$G = 8$	$G = 10$	$G = 12$	$G = 14$	$G = 16$	$G = 18$
4	1.572	1.510	1.389	1.388	1.382	1.381
5	2.064	1.983	1.888	1.855	1.852	1.851
6	2.105	1.806	1.780	1.767	1.764	1.763

3.7.4 Scenario 4 - effect of varying G and c using random hashing

In Scenario 4, we will continue to vary parameter G along with another parameter, the common sample size c. In our proposed method, we assume every node has common sample data points which we use to define Grid. Therefore, we vary both G and c to observe their combined effect on the performance ratios.

We consider $G \in \{8, 10, 12, 14, 16, 18\}$ and $c \in \{4, 5, 6\}$ while keeping other parameter values the same as $N = 20$, $n_i = 500$ and $k = 32$. We use random hashing for generating a univariate sample for our study. We also use *hard* thresholding type and wavelet decomposition level dependency *lev dep* as *true*.

We may refer to the tables 3.8 and 3.9 for studying the results of the variation of G and c on the KS and CVM ratios.

Table 3.9: CVM ratio for varying G and c using random hashing with Normal distribution having $N = 20$, $n_i = 500$, $k = 32$ in Scenario 4

c	$G = 8$	$G = 10$	$G = 12$	$G = 14$	$G = 16$	$G = 18$
4	5.179	4.413	4.071	3.917	3.894	3.884
5	9.454	8.363	8.351	8.305	8.236	8.233
6	6.995	6.640	6.648	6.622	6.624	6.623

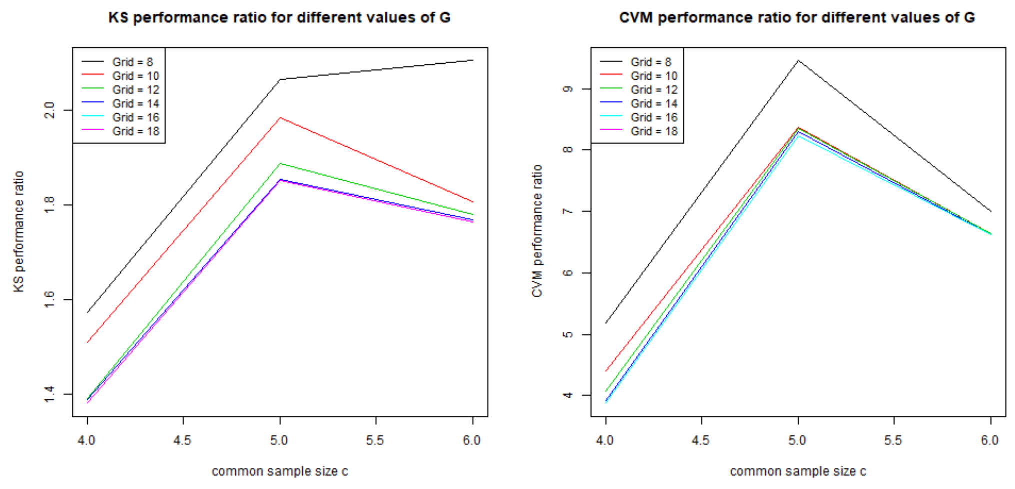

Figure 3.3: Illustration of the effect of varying G and c on the KS and CVM ratios using random hashing with Normal distribution having $N = 20$, $n_i = 500$ and $k = 32$ in scenario 4

We observe that all the KS and CVM ratios are greater than one stipulating the effectiveness of our method in this scenario. The performance ratios, however, decrease with increase in G for a fixed c. The KS and CVM ratio statistics, however, seem to increase as c increases for a fixed G.

From the fig 3.3, we also note that the observed trend is similar for all the values of G except for $G = 8$.

3.7.5 Scenario 5 - effect of varying G using ordered hashing

In Scenario 5, we will vary parameter G between the values $\{8, 10, 12, 14, 16, 18\}$ to observe the effect of variation on the KS and CVM ratios outcome while using ordered hashing for a univariate sample generation. Here, we also expect to observe whether vari-

Table 3.10: KS and CVM ratios for varying G using ordered hashing with normal distribution having $c = 4$, $N = 20$, $n_i = 500$, $k = 32$ in Scenario 5

G	KS performance ratio	CVM performance ratio
8	1.132	1.380
10	0.792	1.250
12	1.282	1.232
14	1.294	1.228
16	1.288	1.226
18	1.286	1.226

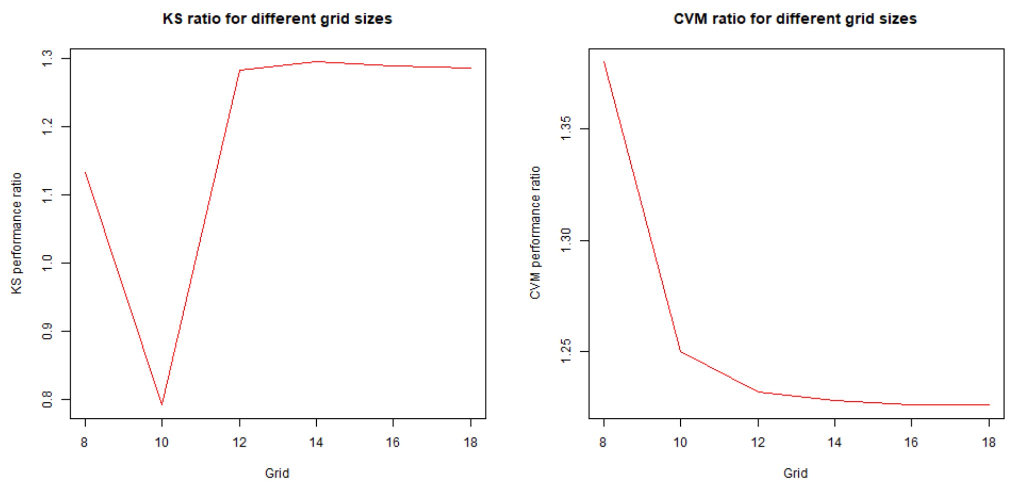

Figure 3.4: Illustration of the effect of varying G on the KS and CVM ratios using random hashing with Normal distribution having $c = 4$, $N = 20$, $n_i = 500$ and $k = 32$ in scenario 5

ation of parameter G affects the outcome in the same manner as varying G with random hashing did.

We consider the parameter values $c = 4$, $N = 20$, $n_i = 500$ and $k = 32$ along with ordered hashing, *hard* thresholding and wavelet decomposition level dependency as *true*.

The table 3.10 shows the results of the variation of G on the KS and CVM ratios with ordered hashing.

As a result of our experiment with scenario 5, we have an indication that the KS and CVM ratios are all greater than one. However, the ratios remains more or less consistent with increase in the value of G. The results do not resemble our findings from scenario 3 where we observed the KS and CVM ratios decreased with increase in values of G.

Table 3.11: KS ratio for varying G and c using ordered hashing with normal distribution having N = 20, n_i = 500, k = 32 in Scenario 6

c	$G = 8$	$G = 10$	$G = 12$	$G = 14$	$G = 16$	$G = 18$
4	1.132	0.792	1.282	1.294	1.288	1.286
5	0.676	0.694	0.681	0.705	0.682	0.682
6	0.695	0.591	0.616	0.634	0.634	0.634

Table 3.12: CVM ratio for varying G and c using ordered hashing with normal distribution having N = 20, n_i = 500, k = 32 in Scenario 6

c	$G = 8$	$G = 10$	$G = 12$	$G = 14$	$G = 16$	$G = 18$
4	1.380	1.250	1.232	1.228	1.226	1.226
5	0.683	0.395	0.378	0.373	0.372	0.372
6	0.809	0.603	0.593	0.594	0.593	0.593

Also, we observed that with random hashing in scenario 3, KS and CVM ratios yielded higher values above one than with ordered hashing. This indicates that ordered hashing may have a stronger effect on the performance ratio than random hashing.

We also observe that there is a sudden drop in the KS performance ratio at G=10 against all other Grid sizes. The scope of our study limits further speculation about the underlying reasons behind such behaviour.

3.7.6 Scenario 6 - effect of varying G and c using ordered hashing

Scenario 6 uses the same approach for data simulation as scenario 5. However, we now vary two parameters at a time to see the combined effect of parameters variation on the KS and CVM ratios.

We consider $G \in \{8, 10, 12, 14, 16, 18\}$ and $c \in \{4, 5, 6\}$ and keep other parameter values $N = 20$, $n_i = 500$ and $k = 32$ with ordered hashing for generating univariate sample of interest. We also use *hard* thresholding type with wavelet decomposition level dependency $lev\ dep$ as $true$.

The tables 3.11 and 3.12 show the results of the variation of G and c on the KS and CVM ratios.

From the results, we find that the values of KS and CVM ratios are greater than one

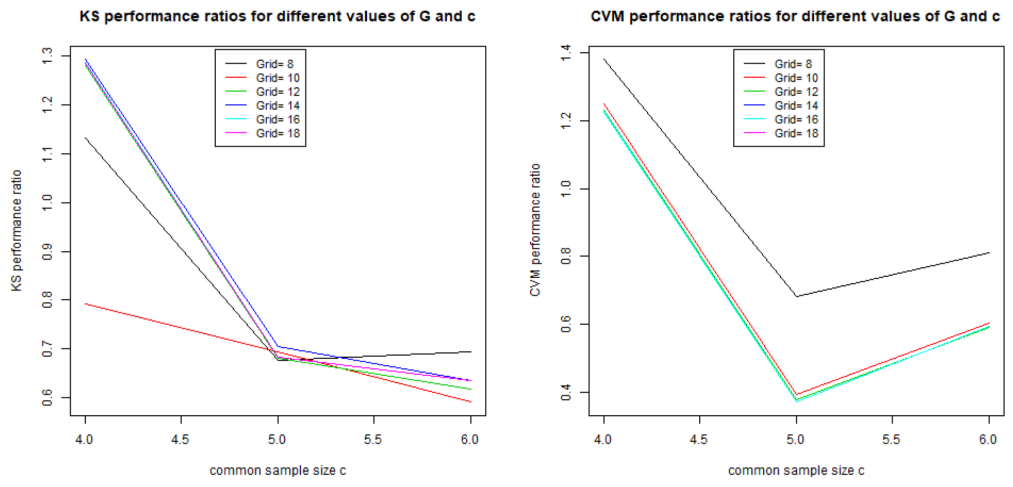

Figure 3.5: Illustration of the effect of varying G and c using ordered hashing with Normal distribution having $N = 20$, $n_i = 500$ and $k = 32$ in scenario 6

for all values of G with $c = 4$ while it declines below one for other values of G for a specific c.

We find for increasing G, the KS and CVM ratios seem to decrease by a small margin while keeping the value of c fixed, similar to scenario 4 where we observed a decline in the performance ratios with random hashing. With increase in c, we observe that the KS and CVM ratios seem to decrease for a fixed G. This trend is unlike scenario 4 where the ratios declined as c increased.

We also observe that values obtained with random hashing were slightly higher than with ordered hashing, indicating a stronger effect of ordered hashing on the KS and CVM performance ratios than random hashing.

Another important observation could be made here about the behavior of G. When G is greater than or equal to 12, the ratios are nearly the same, than the performance of values when $G < 12$. The fig 3.5 shows that the lines are almost converging for values of G greater than 12. This is interesting because a similar trend was also observed with random hashing in scenario 4 while varying G.

We will be now moving on to another set of parameter variation with Normal distribution.

Table 3.13: KS and CVM ratios for varying N using random hashing with normal distribution having $n_i = 500$, $G = 8$, $c = 4$, $k = 32$ in Scenario 7

N	KS performance ratio	CVM performance ratio
20	1.572	5.179
50	1.302	4.915
100	0.816	1.544
250	0.513	0.801

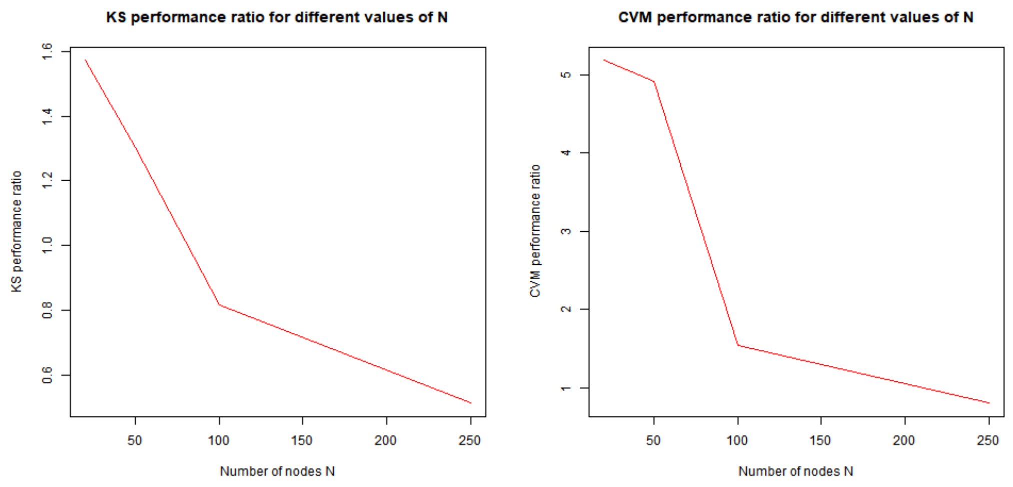

Figure 3.6: Illustration of the effect of varying N using random hashing with Normal distribution having $n_i = 500$, $G = 8$, $c = 4$ and $k = 32$ in scenario 7

3.7.7 Scenario 7 - effect of varying N using random hashing

We tried a number of combinations with parameter variation of G and c in the previous scenarios. We now consider observing the effect of the number of nodes N on the outcome of evaluation measure using KS and CVM ratio statistics.

For scenario 7, we consider varying N between the values of $\{20, 50, 100, 250\}$ with a fixed sample at each node $n_i = 500$. We consider *hard* thresholding type and wavelet decomposition level dependency $lev\ dep$ as $true$. All other parameter values are kept the same with $c = 4$, $G = 8$ and $k = 32$ along with random hashing for generating a univariate sample of interest.

The table 3.13 shows the output of KS and CVM ratio statistics while varying N using random hashing.

Table 3.14: KS ratio for varying N and n_i using random hashing with normal distribution having $G = 8$, $c = 4$, $k = 32$ in Scenario 8

N	$n_i = 500$	$n_i = 2000$	$n_i = 5000$	$n_i = 8000$	$n_i = 10000$
20	1.572	1.773	1.856	1.650	1.533
50	1.302	0.916	1.034	1.243	0.886
100	0.816	0.811	0.630	0.727	0.812
250	0.513	0.505	0.380	0.487	0.390

Table 3.15: CVM ratio for varying N and n_i using random hashing with normal distribution having $G = 8$, $c = 4$, $k = 32$ in Scenario 8

N	$n_i = 500$	$n_i = 2000$	$n_i = 5000$	$n_i = 8000$	$n_i = 10000$
20	5.179	9.494	8.607	8.171	5.575
50	4.915	2.558	3.603	3.561	1.607
100	1.544	1.404	1.017	1.316	1.693
250	0.801	0.610	0.428	0.541	0.440

From the results, we observe that the lower values of N yield performance ratios greater than one or closer to one indicating the effectiveness of our proposed method in this scenario when value of N is low. Hence, the KS and CVM ratios decline with increase in N and eventually fall below one.

3.7.8 Scenario 8 - effect of varying N and n_i using random hashing

Scenario 7 indicated an effect of varying N on the performance ratios. Therefore, in scenario 8, we now consider varying the number of nodes, N and the size of the samples there in, n_i. We vary N between the values $\{20, 50, 100, 250\}$ and n_i between the values $\{500, 2000, 5000, 8000, 10000\}$.

We keep other parameter values the same as $G = 8$, $c = 4$ and $k = 32$ with random hashing for generating a univariate sample. We also use *hard* thresholding type with the wavelet decomposition level dependency *lev dep* as true.

We may refer to the tables 3.14 and 3.15 for the result of varying N and n_i on the KS and CVM ratios using random hashing.

From the results, we observe that there are some values which are much greater than one and also values which are lower than one. For a fixed value of n_i, we find that

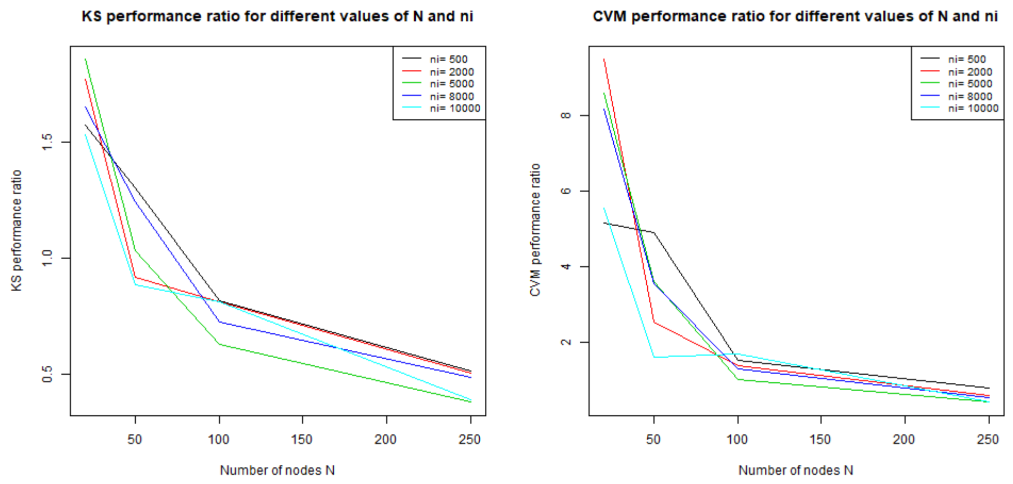

Figure 3.7: Illustration of the effect of varying N and n_i using random hashing with Normal distribution having $G = 8$, $c = 4$ and $k = 32$ in scenario 8

the KS and CVM performance ratios decrease with increase in N. This aligns with our observation from the previous scenario 7. For a fixed value of N, we also find that KS and CVM ratios seem to decrease as n_i increases.

3.7.9 Scenario 9 - effect of varying N using ordered hashing

In Scenario 9, we consider varying N between the values of $\{20, 50, 100, 250\}$ using ordered hashing.

We keep the other parameter values same as previous scenarious with $c = 4$, $N = 20$, $n_i = 500$ and $k = 32$. We use *hard* thresholding with wavelet decomposition level dependency $lev\ dep$ as $true$.

The table 3.16 shows the result of the variation of N on the KS and CVM performance ratios with ordered hashing.

We observe that all the values of KS and CVM performance ratios are above one. We also observe performance ratios remain stable without much change in the values as N increases.

Table 3.16: KS and CVM ratios for varying N using ordered hashing with normal distribution having n_i = 500, G = 8, c = 4, k = 32 in Scenario 9

N	KS performance ratio	CVM performance ratio
20	1.132	1.380
50	1.208	1.022
100	1.205	1.163
250	1.058	1.119

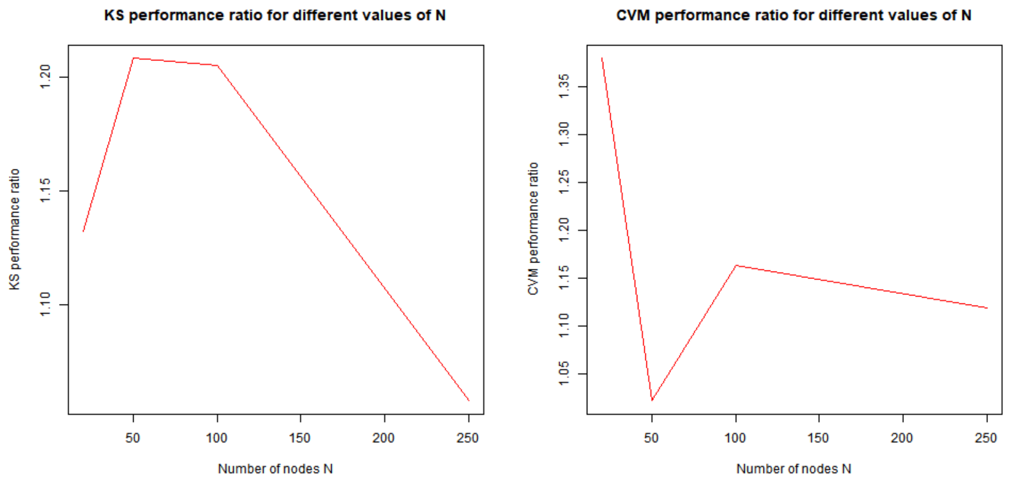

Figure 3.8: Illustration of the effect of varying N using ordered hashing with Normal distribution having n_i = 500, G = 8, c = 4 and k = 32 in scenario 9

3.7.10 Scenario 10 - effect of varying N with n_i using ordered hashing

Scenario 10 uses the same approach for data simulation as scenario 9. However, we now consider varying two parameters at a time to observe their combined effect on the KS and CVM ratios.

We consider $N \in \{20, 50, 100, 250\}$ and $n_i \in \{500, 2000, 5000, 8000, 10000\}$ and keep all other parameter values $G = 8$, $c = 4$ and $k = 32$. We use ordered hashing for generating a univariate sample of interest. We also use *hard* thresholding with dependency on the wavelet decomposition level *lev dep* as *true*.

The tables 3.17 and 3.18 show the results of the variation of N and n_i on the KS and

Table 3.17: KS ratio for varying N and n_i using ordered hashing with normal distribution having $G = 8$, $c = 4$, $k = 32$ in Scenario 10

N	$n_i = 500$	$n_i = 2000$	$n_i = 5000$	$n_i = 8000$	$n_i = 10000$
20	1.132	1.658	1.606	1.430	1.501
50	1.208	1.364	1.102	1.206	1.165
100	1.205	1.431	1.616	1.706	1.732
250	1.058	0.614	0.495	1.391	2.277

Table 3.18: CVM ratio for varying N and n_i using ordered hashing with normal distribution having $G = 8$, $c = 4$, $k = 32$ in Scenario 10

N	$n_i = 500$	$n_i = 2000$	$n_i = 5000$	$n_i = 8000$	$n_i = 10000$
20	1.380	2.696	1.692	2.800	3.836
50	1.022	2.073	1.266	1.455	0.911
100	1.163	0.880	1.839	1.609	1.860
250	1.119	0.602	1.014	1.573	2.159

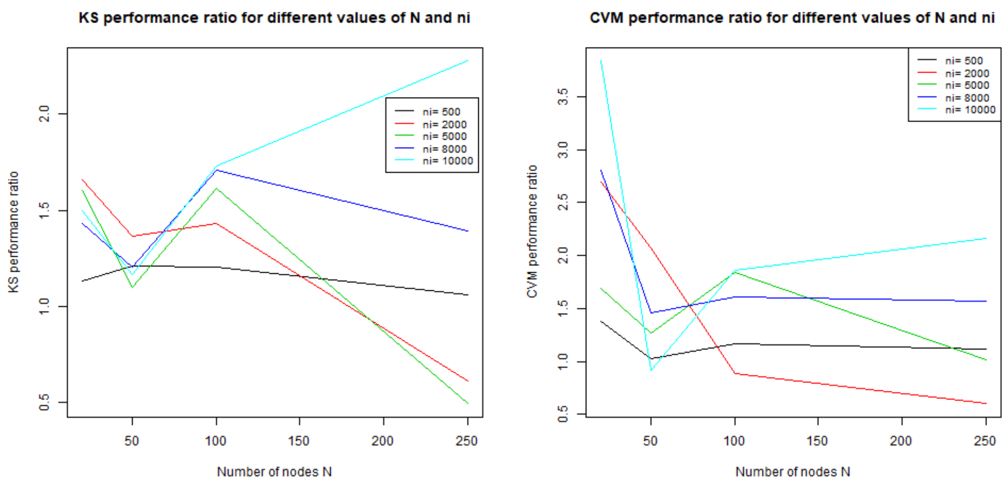

Figure 3.9: Illustration of the effect of varying N and n_i using ordered hashing with Normal distribution having $G = 8$, $c = 4$ and $k = 32$ in scenario 10

CVM ratios.

The outcome of KS and CVM performance ratios declines as N increases for a fixed n_i. This resembles with our observations in scenario 8 where we observed a decreasing trend in KS and CVM ratios for increasing N for a fixed n_i. The ratios, however, seem to increase as n_i increases for a fixed N, unlike results from scenario 8.

We also observe that the values of KS and CVM ratios are greater than one or close to one.

The KS and CVM performance ratios obtained in scenario 10 with ordered hashing are all greater than one or close to one, indicating the effectiveness of our method in this scenario. However, the ratios in scenario 8 with random hashing are comparably on the higher side from scenario 10.

3.8 Cauchy Distribution

3.8.1 Scenario 11 - effect of varying G using random hashing

Similar to experiments in Normal distribution section, we will conduct the same experiments keeping the same parameters and parameter values. In scenario 11, we want to observe the effect of varying a single parameter, the grid resolution G on the KS and CVM ratios using random hashing for generating a univariate data of interest. The effect of varying Grid resolution G on the resulting performance ratio would indicate whether G has any effect on the outcome. We also hope to find if the results in scenario 11 are similar to its corresponding experiment in scenario 3 with Normal distribution.

We vary G between the values $\{8, 10, 12, 14, 16, 18\}$ and keep all other parameter values the same as previous scenarios with $c = 4$, $N = 20$, $n_i = 500$ and $k = 32$ along with random hashing. We consider *hard* thresholding with dependency on the wavelet decomposition level $lev\ dep = true$.

We may refer to the table 3.19 for studying the results of the variation on the KS and CVM ratios.

From the results, we observe that all the ratios are greater than one indicating the effectiveness of our method in this scenario. For a fixed value of c , the KS and CVM ratios decline for higher values of Grid resolution G. The trend resembles our observation from scenario 3 with Normal distribution.

We also observed that with Cauchy distribution, we attained higher values of KS and

Table 3.19: KS and CVM ratios for varying G using random hashing with Cauchy distribution having $c = 4$, $N = 20$, $n_i = 500$, $k = 32$ in Scenario 11

G	KS performance ratio	CVM performance ratio
8	1.722	8.338
10	1.239	6.081
12	1.203	4.808
14	1.104	3.893
16	1.075	4.008
18	0.978	3.266

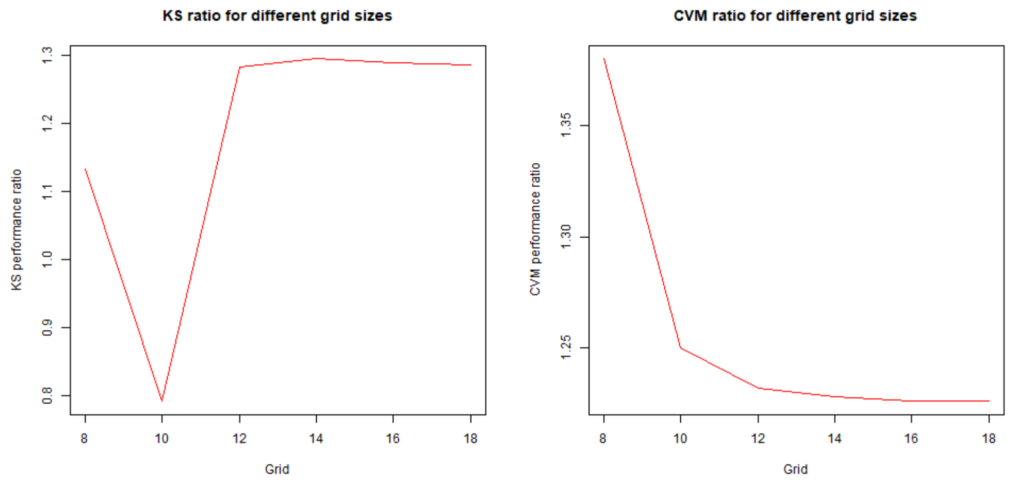

Figure 3.10: Illustration of the effect of varying G using random hashing with Cauchy distribution having $c = 4$, $N = 20$, $n_i = 500$ and $k = 32$ in scenario 11

CVM ratios than with Normal distribution while varying G.

3.8.2 Scenario 12 - effect of varying G and c using random hashing

In Scenario 12, we consider varying parameter G along with the common sample size c to observe their combined effect on the performance ratios and compare with the results of scenario 4 with Normal distribution.

We consider $G \in \{8, 10, 12, 14, 16, 18\}$ and $c \in \{4, 5, 6\}$ while keeping other parameter values $N = 20$, $n_i = 500$ and $k = 32$ with random hashing for data simulation. We also use *hard* thresholding with wavelet decomposition level dependency.

Table 3.20: KS ratio for varying G and c using random hashing with Cauchy distribution having $N = 20$, $n_i = 500$, $k = 32$ in Scenario 12

c	$G = 8$	$G = 10$	$G = 12$	$G = 14$	$G = 16$	$G = 18$
4	1.722	1.239	1.203	1.104	1.075	0.978
5	1.894	1.625	1.517	1.432	1.274	1.180
6	2.110	2.142	2.072	1.956	1.928	1.631

Table 3.21: CVM ratio for varying G and c using random hashing with Cauchy distribution having $N = 20$, $n_i = 500$, $k = 32$ in Scenario 12

c	$G = 8$	$G = 10$	$G = 12$	$G = 14$	$G = 16$	$G = 18$
4	8.338	6.081	4.808	3.893	4.008	3.266
5	7.414	6.020	5.610	5.004	3.756	4.387
6	10.780	9.629	9.635	9.577	9.950	9.429

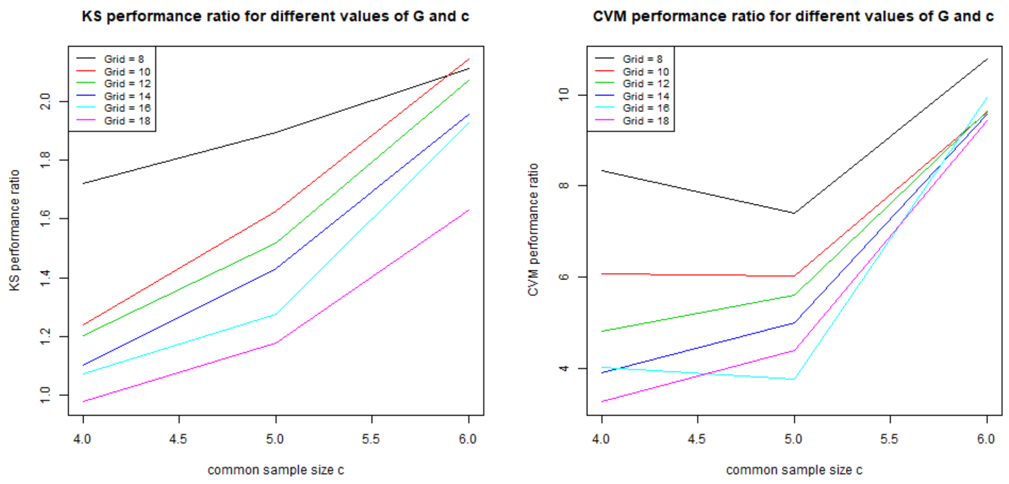

Figure 3.11: Illustration of the effect of varying G and c using random hashing with cauchy distribution having $N = 20$, $n_i = 500$ and $k = 32$ in scenario 12

We may refer to the tables 3.20 and 3.21 for studying the results of the variation of G and c on the KS and CVM ratios.

We observe that all KS and CVM ratios are greater than one. Our findings resemble with the findings from scenario 4 in terms of the effect of G and c variations on the performance ratios. We observe that the ratios decrease with increase in G for a fixed c, similar to our findings in scenario 4 with Normal distribution. However, the effect of varying c for a fixed G on the KS and CVM ratio statistics is not conclusive based on the

Table 3.22: KS and CVM ratios for varying G using ordered hashing with Cauchy distribution having $c = 4$, $N = 20$, $n_i = 500$, $k = 32$ in Scenario 13

G	KS performance ratio	CVM performance ratio
8	1.429	1.346
10	0.834	1.334
12	0.775	1.261
14	0.744	1.140
16	0.751	1.047
18	0.765	1.265

results.

3.8.3 Scenario 13 - effect of varying G using ordered hashing

In scenario 13, we will vary parameter G to observe the effect of variation on the KS and CVM ratios outcome while using ordered hashing for generating a univariate data of interest. We also expect to observe whether variation of parameter G affects the outcome in the same manner as random hashing in scenario 5.

We consider the parameter values $c = 4$, $N = 20$, $n_i = 500$ and $k = 32$, similar to previous scenarios. We use *hard* thresholding and dependency on the wavelet decomposition level $lev\ dep$ as $true$.

For $G \in \{8, 10, 12, 14, 16, 18\}$, we observe the results from table 3.22.

As a result of our experiment with scenario 13, there is an indication that the KS and CVM ratios decreases and falls below one as G increases. Moreover, our findings do not resemble the results from scenario 5 with Normal distribution where we observed the KS and CVM ratios remained stable with increase in G.

We also observed that the trend of performance ratios is similar to the trend observed in scenario 11 with random hashing, where the ratios declined with increase in G. Also, the results are much higher in Cauchy distribution with random hashing in scenario 11 than in scenario 13 with ordered hashing. This indicates that ordered hashing may have a stronger effect on the performance ratio measure than random hashing.

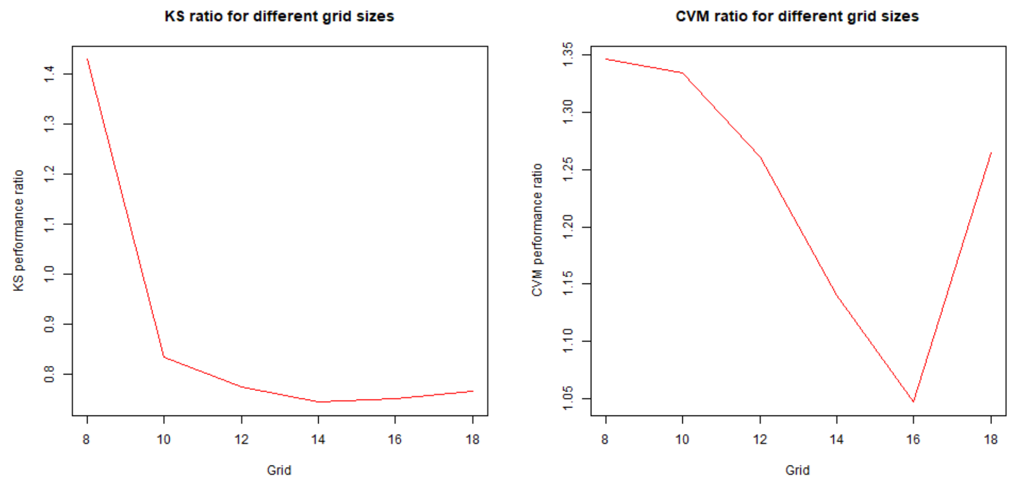

Figure 3.12: Illustration of the effect of varying G using ordered hashing with Cauchy distribution having $c = 4$, $N = 20$, $n_i = 500$ and $k = 32$ in scenario 13

Table 3.23: KS ratio for varying G and c using ordered hashing with Cauchy distribution having $N = 20$, $n_i = 500$, $k = 32$ in Scenario 14

c	$G = 8$	$G = 10$	$G = 12$	$G = 14$	$G = 16$	$G = 18$
4	1.429	0.834	0.775	0.744	0.751	0.765
5	0.795	0.747	0.665	0.657	0.681	0.658
6	0.766	0.731	0.700	0.681	0.665	0.658

3.8.4 Scenario 14 - effect of varying G and c using ordered hashing

The approach for data simulation in scenario 14 is the same as scenario 13. However, we now consider varying two parameters at a time to see the combined effect of parameters variation on the KS and CVM ratios.

We vary $G \in \{8, 10, 12, 14, 16, 18\}$ and $c \in \{4, 5, 6\}$ and keep other parameter values as $N = 20$, $n_i = 500$ and $k = 32$ with ordered hashing for generating a univariate sample of interest. We also use *hard* thresholding along with dependency on the wavelet decomposition level $lev\ dep$ as $true$.

The tables 3.24 and 3.23 show the results of the variation of G and c on the KS and CVM ratios.

From the results, we find that only a few number of values have KS and CVM ratios

Table 3.24: CVM ratio for varying G and c using ordered hashing with Cauchy distribution having $N = 20$, $n_i = 500$, $k = 32$ in Scenario 14

c	$G = 8$	$G = 10$	$G = 12$	$G = 14$	$G = 16$	$G = 18$
4	1.346	1.334	1.261	1.140	1.047	1.265
5	0.497	0.389	0.375	0.374	0.368	0.365
6	0.622	0.412	0.402	0.391	0.377	0.347

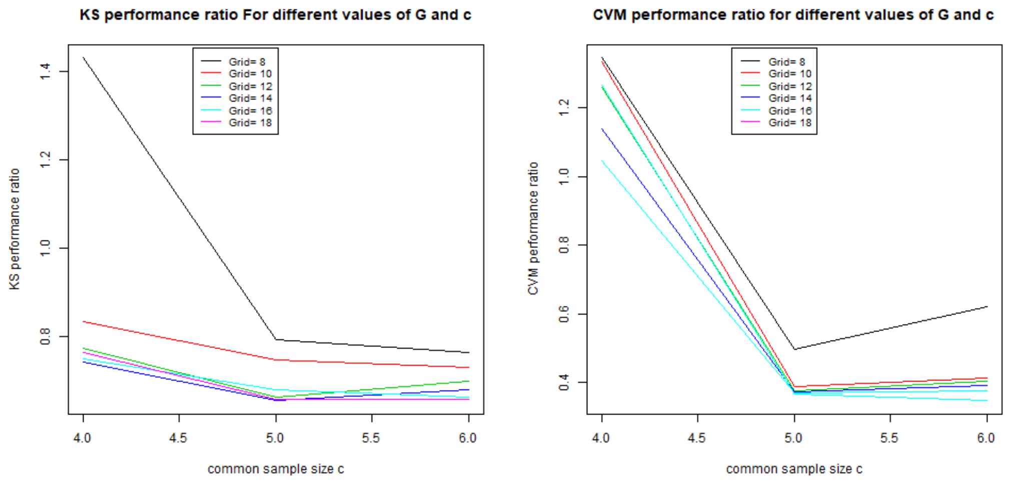

Figure 3.13: Illustration of the effect of varying G and c using ordered hashing with Cauchy distribution having $N = 20$, $n_i = 500$ and $k = 32$ in scenario 14

greater than one while other ratios are below one.

We observe the effect of G on KS and CVM ratios as decreasing while keeping the value of c fixed. This is similar to scenario 12 where we observed a decline in the performance ratios with random hashing. For increase in values of c, we observe that the KS and CVM ratios seem to decrease for a fixed G.

We also observe that values obtained with random hashing were higher than ordered hashing, indicating a stronger effect of ordered hashing on the KS and CVM performance ratios.

Another important observation could be made here from the fig 3.13 that the ratios yield similar values to each other for $G \geq 12$, than the performance of parameter values when $G < 12$. The fig 3.13 shows that the lines are almost converging for values of G greater than 12. This is interesting because a similar trend was also observed with the

Table 3.25: KS and CVM ratios for varying N using random hashing with Cauchy distribution having n_i = 500, c = 4, G = 8, n_i = 500, k = 32 in Scenario 15

G	KS performance ratio	CVM performance ratio
20	1.722	8.338
50	1.168	2.518
100	0.751	0.989
250	0.584	0.700

random hashing in scenario 12 while varying G.

We will be now moving on to another set of parameter variation with Cauchy distribution.

3.8.5 Scenario 15 - effect of varying N using random hashing

We now consider observing the effect of number of nodes N on the outcome of evaluation measure using KS and CVM ratio statistics.

For $N \in \{20, 50, 100, 250\}$ with a fixed sample at each node n_i = 500, we consider *hard* thresholding with the option of wavelet decomposition level for scenario 15. All other parameter values are kept the same with $c = 4$, $G = 8$ and $k = 32$ along with random hashing for generating a univariate sample of interest.

We now observe the individual effect of parameter N on the outcome.

From the results in table 3.25, we find that for lower values of N, the performance ratios are greater than one or closer to one indicating the effectiveness of our proposed method in this scenario except when $N = 250$. This implies that the KS and CVM ratios decrease as N increases and eventually fall below one. This resembles with our observation in scenario 7.

3.8.6 Scenario 16 - effect of varying N and n_i using random hashing

The results from scenario 15 indicated an effect of varying N on the KS and CVM performance ratios. Therefore, in scenario 16, we now consider varying the number of nodes, N and the size of the samples there in, n_i. For $N \in \{20, 50, 100, 250\}$ and $n_i \in$

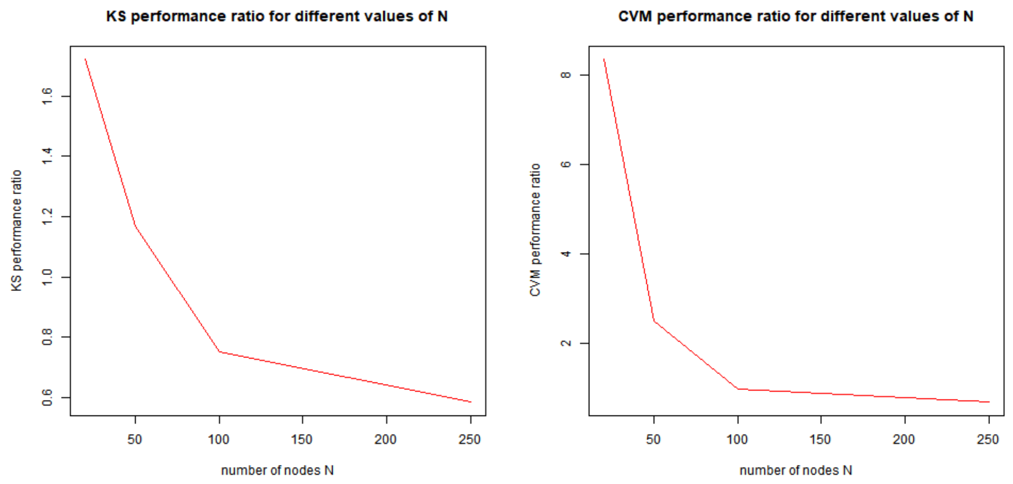

Figure 3.14: Illustration of the effect of varying N using random hashing with Cauchy distribution having $n_i = 500$, $c = 4$, $G = 8$, $n_i = 500$ and $k = 32$ in scenario 15

Table 3.26: KS ratio for varying N and n_i using random hashing with Cauchy distribution having $c = 4$, $G = 8$, $k = 32$ in Scenario 16

N	$n_i = 500$	$n_i = 2000$	$n_i = 5000$	$n_i = 8000$	$n_i = 10000$
20	1.722	1.276	1.488	1.478	1.867
50	1.168	1.267	1.115	0.899	0.943
100	0.751	0.589	0.604	0.609	0.739
250	0.584	0.395	0.492	0.408	0.550

Table 3.27: CVM ratio for varying N and n_i using random hashing with Cauchy distribution having $c = 4$, $G = 8$, $k = 32$ in Scenario 16

N	$n_i = 500$	$n_i = 2000$	$n_i = 5000$	$n_i = 8000$	$n_i = 10000$
20	8.338	5.031	5.659	6.112	4.285
50	2.518	3.292	3.028	4.022	1.449
100	0.989	1.065	0.583	1.446	1.454
250	0.700	0.488	0.456	0.593	0.528

$\{500, 2000, 5000, 8000, 10000\}$, we consider keeping other parameter values $G = 8$, $c = 4$ and $k = 32$ with random hashing for generating a univariate data of interest. We also use *hard* thresholding with dependency on the wavelet decomposition level.

We may refer to the tables 3.26 and 3.27 for the result of varying N and n_i on the KS and CVM ratios using random hashing.

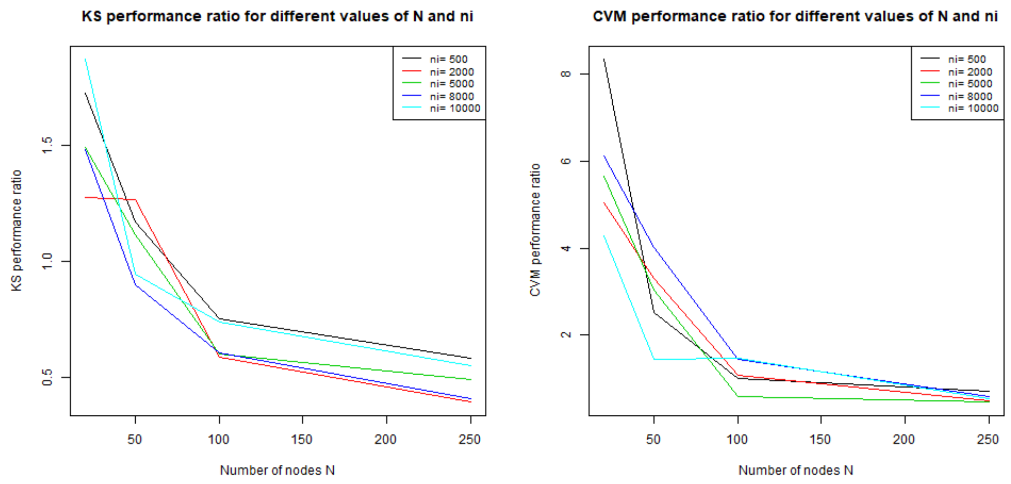

Figure 3.15: Illustration of the effect of varying N and n_i using random hashing with Cauchy distribution having $c = 4$, $G = 8$ and $k = 32$ in scenario 16

From the results, we find that not all performance ratios are greater than one. The ratios are below one for higher values of N for a particular n_i. We also observe that for a fixed n_i, the performance ratios decrease with increase in N. Also, the performance ratios seem to decline as n_i increases for a fixed N.

3.8.7 Scenario 17 - effect of varying N using ordered hashing

In Scenario 17, we consider varying N between the values of $\{20, 50, 100, 250\}$ using ordered hashing.

We keep the other parameter values $c = 4$, $N = 20$, $n_i = 500$ and $k = 32$ fixed. We also use *hard* thresholding with the dependency on the wavelet decomposition level.

The table 3.28 shows the results of the variation of N on the KS and CVM performance ratios with ordered hashing in Cauchy distribution.

From the results, we observe the KS and CVM performance ratios are greater than one or almost close to one. We also observe that similar to scenario 15, the performance ratios decrease with increase in N.

Table 3.28: KS and CVM ratios for varying N using ordered hashing with Cauchy distribution having $n_i = 500$, $c = 4$, $G = 8$, $k = 32$ in Scenario 17

N	KS performance ratio	CVM performance ratio
20	1.429	1.346
50	1.064	1.179
100	1.126	0.893
250	0.978	0.986

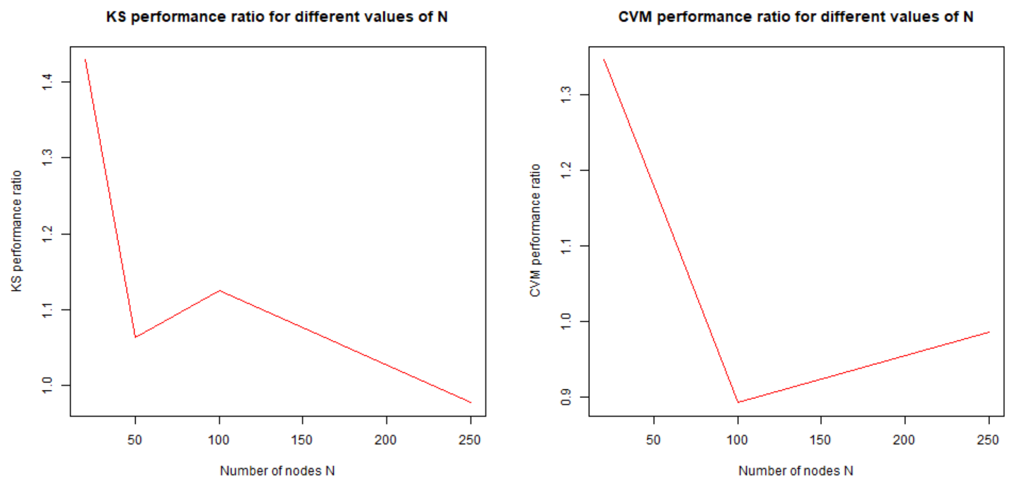

Figure 3.16: Illustration of the effect of varying N using ordered hashing with Cauchy distribution having $n_i = 500$, $c = 4$, $G = 8$ and $k = 32$ in scenario 17

3.8.8 Scenario 18 - effect of varying N with n_i using ordered hashing

Scenario 18 uses the same approach for generating a univariate data of interest as scenario 17. However, we now consider varying two parameters at a time to observe their combined effect on the KS and CVM ratios.

We consider $N \in \{20, 50, 100, 250\}$ and $n_i \in \{500, 2000, 5000, 8000, 10000\}$ and keep all other parameter values the same as $G = 8$, $c = 4$ and $k = 32$ with ordered hashing for generating a univariate sample. We also use *hard* thresholding with dependency on the wavelet decomposition level *lev dep* as *true*.

The tables 3.29 and 3.30 show the results of the variation of N and n_i on the KS and CVM ratios.

We observe that the values of KS and CVM ratios are greater than one or closer to one

Table 3.29: KS ratio for varying N and n_i using ordered hashing with Cauchy distribution having $c = 4$, $G = 8$, $k = 32$ in Scenario 18

N	$n_i = 500$	$n_i = 2000$	$n_i = 5000$	$n_i = 8000$	$n_i = 10000$
20	1.429	1.382	1.063	1.386	1.200
50	1.064	1.235	1.083	1.091	1.441
100	1.126	1.093	1.152	0.813	1.128
250	0.978	0.939	1.278	0.663	0.501

Table 3.30: CVM ratio for varying N and n_i using ordered hashing with Cauchy distribution having $c = 4$, $G = 8$, $k = 32$ in Scenario 18

N	$n_i = 500$	$n_i = 2000$	$n_i = 5000$	$n_i = 8000$	$n_i = 10000$
20	1.346	2.188	1.949	2.342	1.757
50	1.179	1.599	1.313	2.296	1.201
100	0.893	1.148	1.462	0.676	1.282
250	0.986	0.935	1.460	0.373	0.259

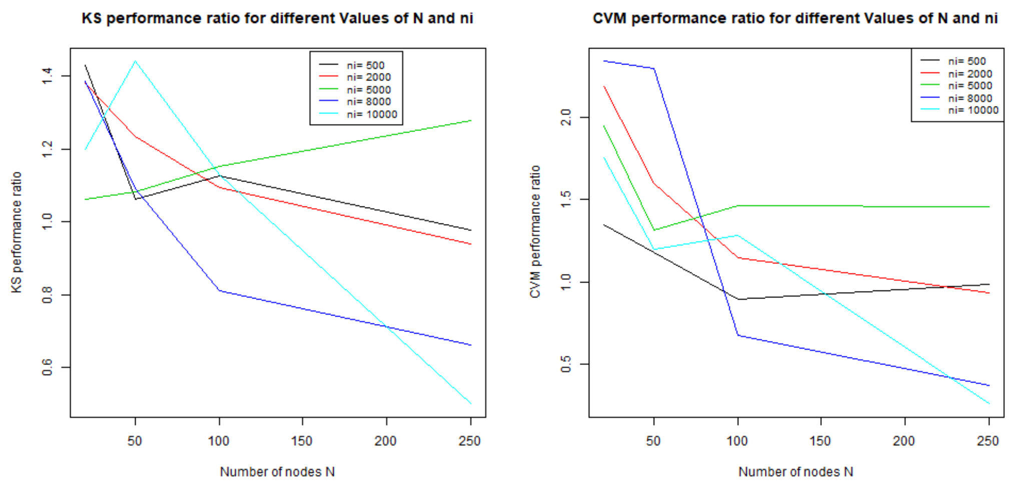

Figure 3.17: Illustration of the effect of varying N and n_i using ordered hashing with Cauchy distribution having $c = 4$, $G = 8$ and $k = 32$ in scenario 18

for some values indicating the effectiveness of our method in this scenario.

There seems to be a decreasing trend as N increases for a fixed n_i. This is similar to scenario 16 where KS and CVM ratios decreased as N increased for a fixed n_i. We also observe that the KS and CVM performance ratios seem to increase as n_i increases for a fixed N.

3.9 Gamma Distribution

In this section, we will conduct experiments based on what we have learned from previous two distribution sections (Normal and Cauchy). We will explore fewer combinations of parameters and parameter values in this section.

So far, we have observed that KS and CVM ratios display a decreasing trend while varying parameter G and N. Both parameter variations have the same effect irrespective of using random hashing or ordered hashing for generating a univariate data of interest for our study.

For the next set of simulations, we make a subjective decision based on our observations to not consider single parameter variations. However, we are still interested in observing the effect of varying two parameters at a time on the KS and CVM performance ratios.

Also, because of close performance in KS and CVM ratios for values of G above 12 in previous two distributions, we will consider varying G between the values $\{8, 10, 12\}$ for both random hashing and ordered hashing method of simulating our univariate data of interest.

We now begin varying parameters G and c in scenario 19 to observe their combined effects.

Table 3.31: KS ratio for varying G and c using random hashing with Gamma distribution having N = 20, n_i = 500, k = 32 in Scenario 19

c	G= 8	G= 10	G= 12
4	1.674	1.530	1.462
5	1.950	1.820	1.692
6	2.438	2.235	2.209

Table 3.32: CVM ratio for varying G and c using random hashing with Gamma distribution having N = 20, n_i = 500, k = 32 in Scenario 19

c	G= 8	G= 10	G= 12
4	4.663	4.711	4.648
5	6.863	6.535	6.070
6	8.317	7.499	7.318

3.9.1 Scenario 19 - effect of varying G and c using random hashing

In Scenario 19, we consider varying parameter G along with the common sample size c to observe their combined effect on the performance ratios and compare with the results of scenarios 4 and 12 in Normal and Cauchy distributions respectively.

We consider $G \in \{8, 10, 12\}$ and $c \in \{4, 5, 6\}$ while keeping other parameter values as $N = 20$, $n_i = 500$ and $k = 32$. We use random hashing for generating univariate data of interest. We also use *hard* thresholding with wavelet decomposition level dependency.

We may refer to the tables 3.31 and 3.32 for studying the results of the variation of G and c on the KS and CVM performance ratio.

We observe that all KS and CVM ratios are greater than one indicating the effectiveness of our method. Our findings resemble with the results obtained in scenario 4 and 12. We observe that the performance ratios decrease as G increases for a fixed c. As c increases, the KS and CVM ratios seem to increase as well.

3.9.2 Scenario 20 - effect of varying G and c using ordered hashing

The approach for data simulation in scenario 20 is the same as scenario 19. However, we now consider varying two parameters using ordered hashing for generating univariate

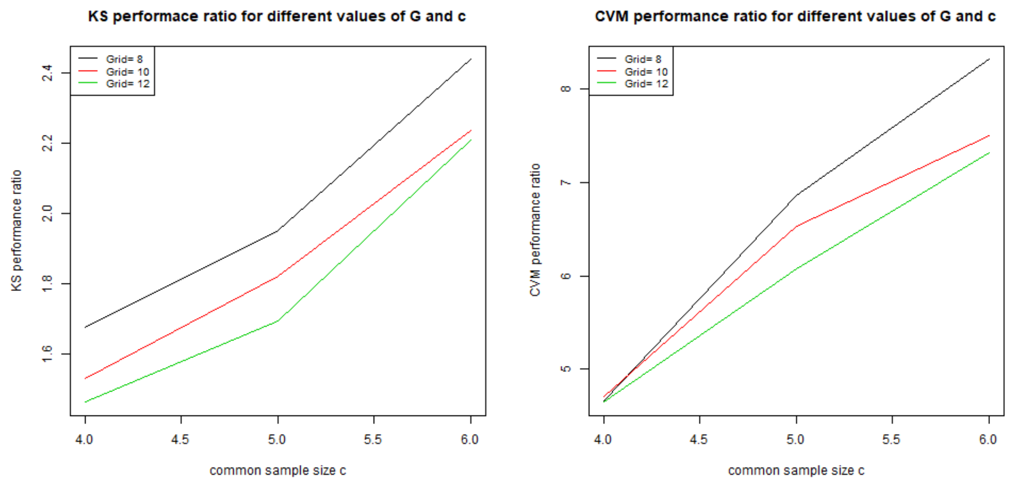

Figure 3.18: Illustration of the effect of varying G and c using random hashing with Gamma distribution having $N = 20$, $n_i = 500$ and $k = 32$ in scenario 19

Table 3.33: KS ratio for varying G and c using ordered hashing with Gamma distribution having $N = 20$, $n_i = 500$, $k = 32$ in Scenario 20

c	$G = 8$	$G = 10$	$G = 12$
4	1.358	1.065	1.312
5	0.971	0.771	0.754
6	0.655	0.602	0.584

Table 3.34: CVM ratio for varying G and c using ordered hashing with Gamma distribution having $N = 20$, $n_i = 500$, $k = 32$ in Scenario 20

c	$G = 8$	$G = 10$	$G = 12$
4	1.298	1.426	1.321
5	0.593	0.432	0.432
6	0.672	0.459	0.448

data of interest.

We consider $G \in \{8, 10, 12\}$ and $c \in \{4, 5, 6\}$ and keep other parameter values the same as $N = 20$, $n_i = 500$ and $k = 32$. We also use *hard* thresholding along with dependency on the wavelet decomposition level.

The tables 3.33 and 3.34 show the results of the varying G and c on the KS and CVM ratios.

From the results, we find only a few KS and CVM ratios greater than one while most

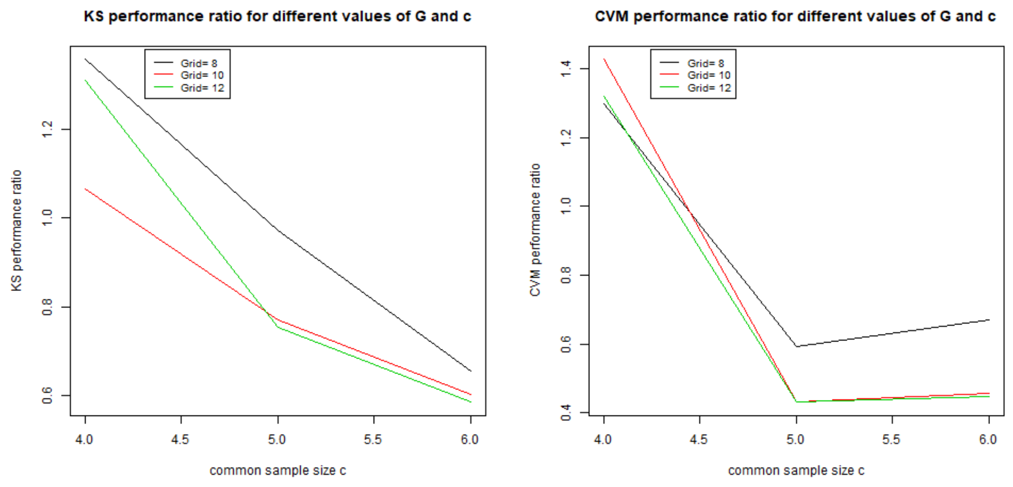

Figure 3.19: Illustration of the effect of varying G and c using ordered hashing with Gamma distribution having $N = 20$, $n_i = 500$ and $k = 32$ in scenario 20

ratios are obsrved below one.

We observe that the KS and CVM ratios seem to be decreasing as G increases. However, unlike scenario 19, as c increases, the performance ratios based on KS and CVM statistics also decrease.

We also observe that the ratios obtained with random hashing were higher than ordered hashing, indicating a stronger effect of ordered hashing on the KS and CVM performance ratios.

We will be now moving on to another set of parameter variation with Cauchy distribution.

3.9.3 Scenario 21 - effect of varying N and n_i using random hashing

In scenario 21, we now consider varying the number of nodes, N and the size of the samples there in, n_i. We vary N between the values $\{20, 50, 100, 250\}$ and n_i between the values $\{500, 2000, 5000, 8000, 10000\}$.

We consider keeping other parameter values as $G = 8$, $c = 4$ and $k = 32$. We use random hashing for generating univariate data of interest. We also use *hard* thresholding

Table 3.35: KS ratio for varying N and n_i using random hashing with Gamma distribution having $G = 8$, $c = 4$, $k = 32$ in Scenario 21

N	$n_i = 500$	n_i= 2000	n_i= 5000	n_i= 8000	n_i= 10000
20	1.674	1.934	1.494	1.749	1.581
50	1.303	1.066	0.964	1.138	0.878
100	1.134	0.828	0.637	0.731	0.733
250	0.620	0.563	0.479	0.476	0.432

Table 3.36: CVM ratio for varying N and n_i using random hashing with Gamma distribution having $G = 8$, $c = 4$, $k = 32$ in Scenario 21

N	$n_i = 500$	$n_i = 2000$	$n_i = 5000$	$n_i = 8000$	$n_i = 10000$
20	4.663	6.573	5.963	7.457	9.305
50	5.616	2.462	2.071	3.232	1.834
100	3.298	1.351	1.225	1.370	0.988
250	1.080	0.801	0.463	0.413	0.549

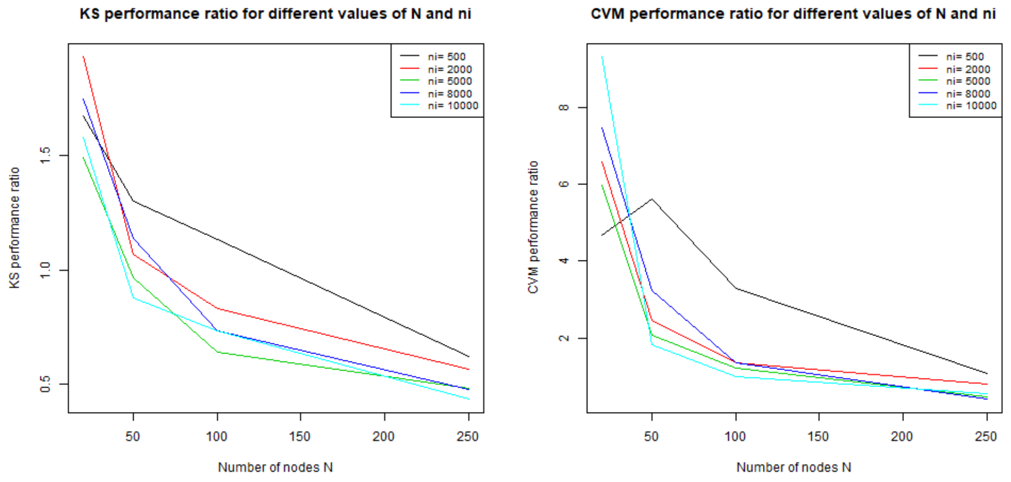

Figure 3.20: Illustration of the effect of varying N and n_i using ordered hashing with Gamma distribution having $G = 8$, $c = 4$ and $k = 32$ in scenario 21

with the wavelet decomposition level dependency.

We may refer to the tables 3.35 and 3.36 for the observing the effect of varying N and n_i on the KS and CVM ratios using random hashing.

From the results, we find that performance ratios based on KS and CVM statistics are greater than one or closer to one for lower values of N.

Table 3.37: KS ratio for varying N and n_i using ordered hashing with Gamma distribution having $G = 8$, $c = 4$, $k = 32$ in Scenario 22

N	$n_i = 500$	$n_i = 2000$	$n_i = 5000$	$n_i = 8000$	$n_i = 10000$
20	1.358	1.889	1.739	1.781	1.846
50	1.137	1.850	2.570	2.459	2.559
100	1.002	1.483	2.063	2.520	2.042
250	1.041	1.295	1.563	1.682	1.811

Table 3.38: CVM ratio for varying N and n_i using ordered hashing with Gamma distribution having $G = 8$, $c = 4$, $k = 32$ in Scenario 22

N	$n_i = 500$	$n_i = 2000$	$n_i = 5000$	$n_i = 8000$	$n_i = 10000$
20	1.298	1.694	3.096	7.495	5.086
50	1.316	1.768	2.419	3.542	5.352
100	1.078	1.407	1.947	2.258	2.126
250	1.147	1.251	1.608	1.522	2.026

We also observe that for a fixed n_i, the performance ratios decrease with increase in N. However, the performance ratios seem to decrease while increasing n_i for a fixed N. This resembles with our findings from scenario 16 in Cauchy distribution and from scenario 8 in Normal distribution.

3.9.4 Scenario 22 - effect of varying N and n_i using ordered hashing

For scenario 22, we now consider varying two parameters at a time to observe their combined effect on the KS and CVM ratios.

We consider $N \in \{20, 50, 100, 250\}$ and $n_i \in \{500, 2000, 5000, 8000, 10000\}$ and keep all other parameter values the same as $G = 8$, $c = 4$ and $k = 32$. We use ordered hashing for generating univariate sample of interest. We also use *hard* thresholding with dependency on the wavelet decomposition level.

The tables 3.37 and 3.38 show the results of the variation of N and n_i on the KS and CVM ratios.

In table 3.37 and 3.38, we observe that all KS and CVM performance ratios are greater than one, suggesting the remarkable effectiveness of our proposed method in this scenario. This is unlike our observation in scenario 21 where not all ratios were above one.

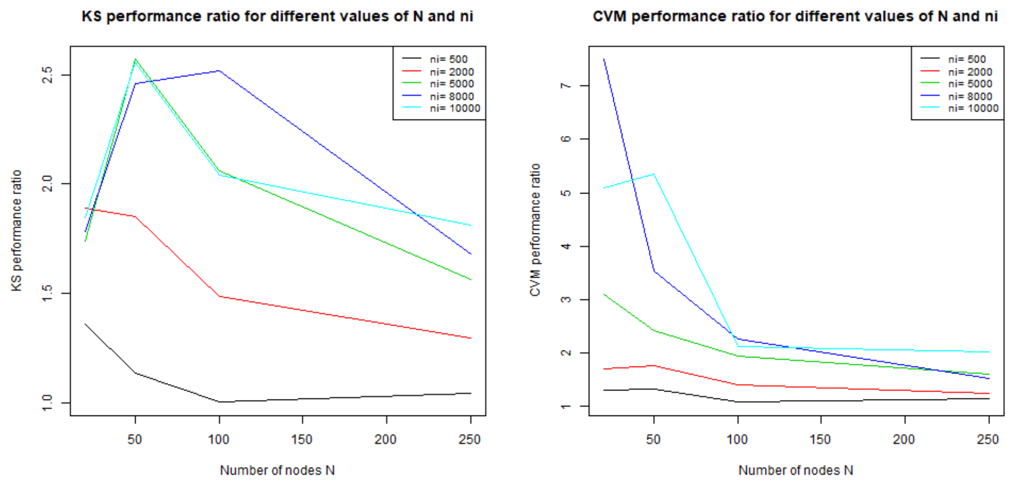

Figure 3.21: Illustration of the effect of varying N and n_i using ordered hashing with Gamma distribution having $G = 8$, $c = 4$ and $k = 32$ in scenario 22

For a fixed n_i, the performance ratios seems to decline as N increases. This is similar to our observation in scenario 21 with random hashing. However, the KS and CVM ratios increase as n_i increases for a fixed N. This observation is unlike our findings in scenario 21, where the ratios seemed to be decreasing as n_i increased for a fixed N.

Unlike our previous observation in Cauchy distribution in scenario 18 but similar to our observation in Normal distribution in scenario 10, we find that the ratios obtained with ordered hashing in scenario 22 are comparably higher than the ratios with random hashing in scenario 21.

3.10 Mixture of Normals Distribution

In this section, we conduct the same experiments as in Gamma distribution section. We are still interested in observing the effect of varying two parameters at a time on the KS and CVM performance ratios. We consider the same parameters and parameter values.

We now begin varying parameters G and c in scenario 23 to observe their combined effects on the performance ratios.

Table 3.39: KS ratio for varying G and c using random hashing with Mixture of Normals having $N = 8$, $n_i = 500$, $k = 32$ in Scenario 23

c	$G = 8$	$G = 10$	$G = 12$
4	1.032	1.030	1.033
5	1.004	1.002	0.996
6	1.018	1.016	1.022

Table 3.40: CVM ratio for varying G and c using random hashing with Mixture of Normals having $N = 8$, $n_i = 500$, $k = 32$ in Scenario 23

c	$G = 8$	$G = 10$	$G = 12$
4	1.083	1.078	1.079
5	1.046	1.051	1.051
6	1.053	1.072	1.079

3.10.1 Scenario 23 - effect of varying G and c using random hashing

In scenario 23, we consider varying G and c and observe their combined effect on the performance ratios. For $G \in \{8, 10, 12\}$ and $c \in \{4, 5, 6\}$, we keep the other parameter values as $N = 20$, $n_i = 500$ and $k = 32$. We use random hashing for generating univariate data of interest. We also use *hard* thresholding with wavelet decomposition level dependency.

The tables 3.39 and 3.40 show the results of the variation.

We observe that all the ratios are greater than one, indicative of the effectiveness of our method in this scenario. However, unlike our previous observations with Normal, Cauchy and Gamma distributions, the results from this experiment showed consistent KS and CVM ratios without much variation as G increases. The same consistency is observed in the performance ratios as c increases.

Similar to Models with other distributions, we observe a rise in measurement of KS and CVM statistics with increase in c. However, it seems with increase in G value, the KS and CVM measure also increases or remains the same throughout.

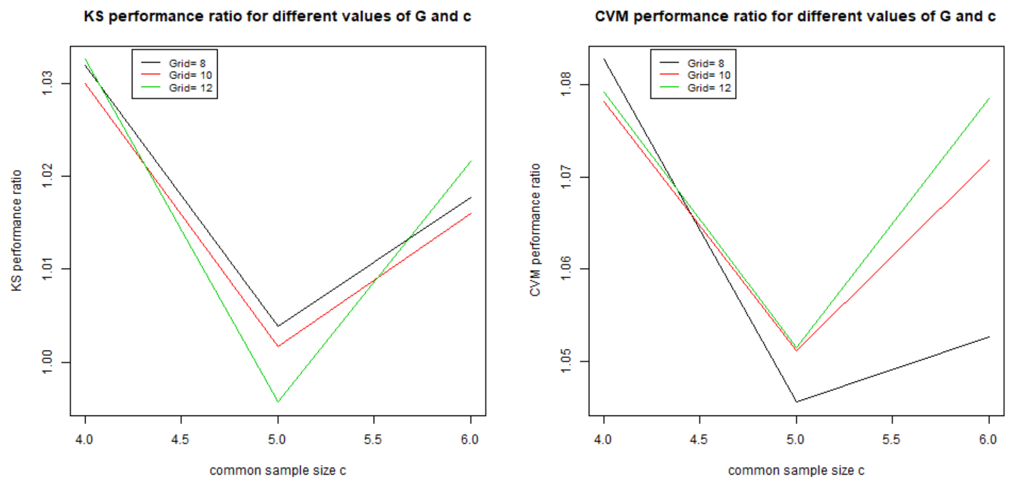

Figure 3.22: Illustration of the effect of varying G and c using random hashing with Mixture of Normals having $N = 20$, $n_i = 500$ and $k = 32$ in scenario 23

Table 3.41: KS ratio for varying G and c using ordered hashing with Mixture of Normals having $N = 8$, $n_i = 500$, $k = 32$ in Scenario 24

c	$G = 8$	$G = 10$	$G = 12$
4	1.009	1.006	1.006
5	0.992	0.990	0.989
6	0.990	0.985	0.982

Table 3.42: CVM ratio for varying G and c using ordered hashing with Mixture of Normals having $N = 8$, $n_i = 500$, $k = 32$ in Scenario 24

c	$G = 8$	$G = 10$	$G = 12$
4	1.001	0.998	0.998
5	0.986	1.001	1.004
6	0.999	1.003	1.008

3.10.2 Scenario 24 - effect of varying G and c using ordered hashing

Similar to scenario 23, we consider varying G and c together,but with ordered hashing this time. We keep the same parameters and parameter values for our experiment. We can see the results in tables $3.41 and 3.42.

In this scenario, we observe all values are greater than one or almost close to one, similar to scenario 23. The performance ratios seem to be more or less stable as G increases

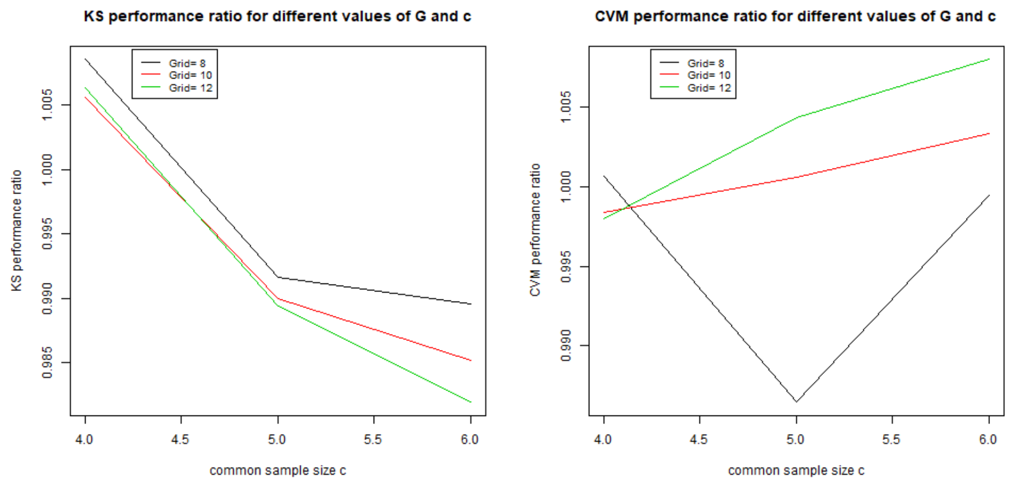

Figure 3.23: Illustration of the effect of varying G and c using ordered hashing with Mixture of Normals having $N = 20$, $n_i = 500$ and $k = 32$ in scenario 24

for a fixed c. We also find similar behavior of consistency on the performance ratios upon increasing c for a fixed G. This resembles with our findings from scenario 23 with random hashing.

3.10.3 Scenario 25 - effect of varying N and n_i using random hashing

In scenario 25, we consider varying the number of nodes, N and the size of the samples there in, n_i. We vary N between the values $\{20, 50, 100, 250\}$ and n_i between the values $\{500, 2000, 5000, 8000, 10000\}$.

We consider keeping other parameter values as $G = 8$, $c = 4$ and $k = 32$. We use random hashing for generating univariate data of interest. We also use *hard* thresholding with the wavelet decomposition level dependency.

We may refer to the tables 3.43 and 3.44 for results.

We observe all the ratios are greater than one. The ratios are stable as N increases for a fixed G. Also, the ratios remain stable as n_i inceases for a fixed N.

Table 3.43: KS ratio for varying N and n_i using random hashing with Mixture of Normals having $G = 8$, $c = 4$, $k = 32$ in Scenario 25

N	$n_i = 500$	$n_i = 2000$	$n_i = 5000$	$n_i = 8000$	$n_i = 10000$
20	1.032	1.017	1.029	1.037	1.013
50	1.011	1.005	1.002	1.009	1.010
100	1.003	1.006	1.004	0.997	1.005
250	0.998	0.991	0.998	0.995	0.990

Table 3.44: CVM ratio for varying N and n_i using random hashing with Mixture of Normals having $G = 8$, $c = 4$, $k = 32$ in Scenario 25

N	$n_i = 500$	$n_i = 2000$	$n_i = 5000$	$n_i = 8000$	$n_i = 10000$
20	1.083	1.023	1.063	1.036	1.049
50	1.000	1.018	0.991	1.019	1.002
100	1.011	1.010	1.017	1.008	1.018
250	1.019	1.002	1.004	1.012	1.003

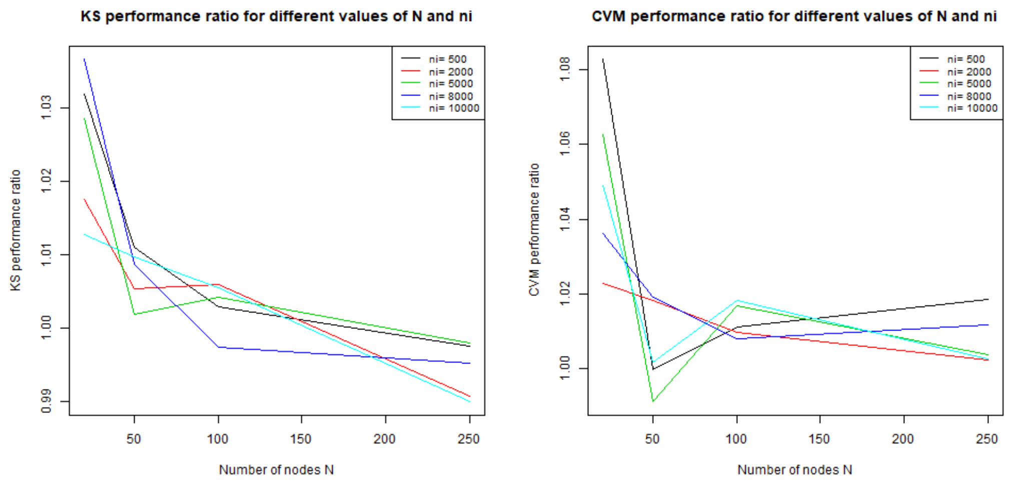

Figure 3.24: Illustration of the effect of varying N and n_i using random hashing with Mixture of Normals having $G = 8$, $c = 4$ and $k = 32$ in scenario 25

Table 3.45: KS ratio for varying N and n_i using ordered hashing with Mixture of Normals having $G = 8$, $c = 4$, $k = 32$ in Scenario 26

N	$n_i = 500$	$n_i = 2000$	$n_i = 5000$	$n_i = 8000$	$n_i = 10000$
20	1.009	1.011	1.009	1.012	1.007
50	1.004	1.005	1.004	1.004	1.004
100	1.001	1.002	1.002	1.003	1.003
250	1.001	1.000	1.000	1.001	1.001

Table 3.46: CVM ratio for varying N and n_i using ordered hashing with Mixture of Normals having $G = 8$, $c = 4$, $k = 32$ in Scenario 26

N	$n_i = 500$	$n_i = 2000$	$n_i = 5000$	$n_i = 8000$	$n_i = 10000$
20	1.001	1.014	0.998	0.990	0.989
50	0.998	1.001	0.996	0.998	0.997
100	1.001	1.000	0.999	0.999	0.998
250	1.000	0.999	1.000	0.999	1.000

3.10.4 Scenario 26 - effect of varying N and n_i using ordered hashing

For scenario 26, we consider varying the same two parameters, N and n_i, similar to scenario 25, with ordered hashing to observe their combined effect on the KS and CVM ratios.

For $N \in \{20, 50, 100, 250\}$ and $n_i \in \{500, 2000, 5000, 8000, 10000\}$, we keep all other parameter values the same as $G = 8$, $c = 4$ and $k = 32$. We use ordered hashing for generating univariate sample of interest. We also use *hard* thresholding with dependency on the wavelet decomposition level.

The tables 3.45 and 3.46 show the results of the variation of N and n_i on the KS and CVM ratios.

Once again, we observe the KS and CVM statistics yield performance ratios almost close to one. The ratios indicate that our proposed method to estimate a wavelet-based EDF by compression strategy is effective in this scenario.

For a fixed n_i, the ratios seem to remain stable as N increases. Also, the ratios are consistent when n_i increases for a fixed N.

We can make another observation here that, unlike other distributions, the ratios are not much higher for random hashing from ordered hashing in Mixture of Normals.

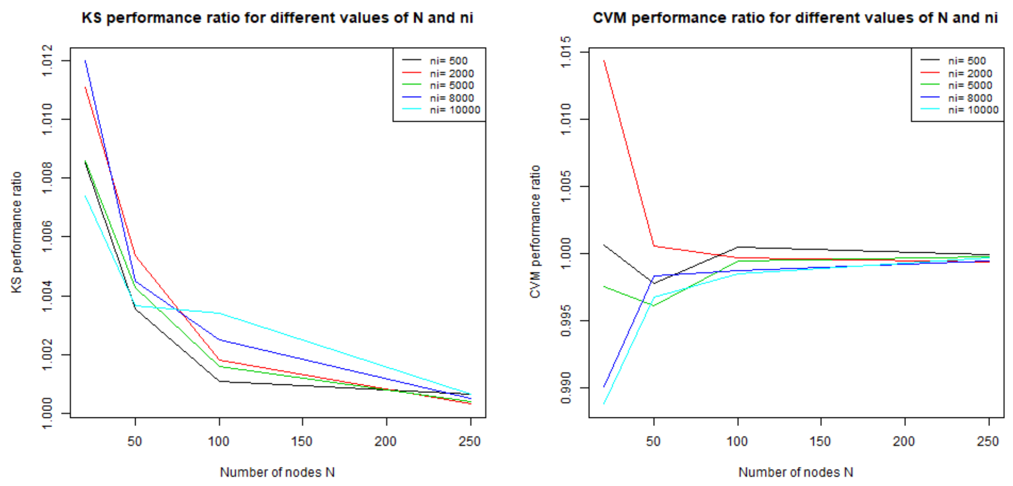

Figure 3.25: Illustration of the effect of varying N and n_i using ordered hashing with Mixture of Normals having $G = 8$, $c = 4$ and $k = 32$ in scenario 26

Now that we have completed our list of simulations, we may refer to the table 3.47 displaying if the KS and CVM performance ratios are above one or below one across all distributions.

Table 3.47: Outcomes of KS and CVM ratios across all distributions

Scenarios	Distributions	Favorable	Unfavorable	Mixed
1	Normal	$\times$		
2	Normal			$\times^*$
3	Normal	$\times$		
4	Normal	$\times$		
5	Normal			$\times^*$
6	Normal			$\times$
7	Normal			$\times$
8	Normal			$\times$
9	Normal	$\times$		
10	Normal	$\times$		

Scenarios	Distributions	Favorable	Unfavorable	Mixed
11	Cauchy	$\times$		
12	Cauchy	$\times$		
13	Cauchy			$\times$
14	Cauchy			$\times$
15	Cauchy			$\times$
16	Cauchy			$\times$
17	Cauchy			$\times^*$
18	Cauchy			$\times^*$
19	Gamma	$\times$		
20	Gamma			$\times$
21	Gamma			$\times$
22	Gamma	$\times$		
23	Mixture	$\times$		
24	Mixture	$\times$		
25	Mixture	$\times$		
26	Mixture	$\times$		

$\times^*$ = the ratios are mostly above one, with a few below one.

We may note from the experiments that some of the scenarios may yield KS and CVM ratios much above one and some may not. The table 3.47 displays favorable result if hundred% of performance ratios are above one, unfavorable if hundred% of performance ratios are below one and mixed if the ratios are both above one and below one. For some scenarios, we may find that ratios are very close to one, and hence, considered as a favorable result.

We will discuss our interpretations from the performed experiments in the next section.

3.11 Discussion of Results

Through out our experiments, we maintained similar conditions of experimentation varying only a few parameters at a time. We conducted independent and combined simulations while keeping the conditions of thresholding (threshold type and wavelet decomposition level dependency) and SRS constant throughout.

Monte Carlo simulation allowed us to check for effectiveness of our method while varying a few parameters. Many scenarios resulted in favorable results where the KS and CVM performance ratios were greater than one. We also identified a trend of the effect of parameters variation using KS and CVM ratio statistics for the parameters that we varied. We studied the observations and interpreted the results as below.

1. Hard thresholding yields better results than soft thresholding. Hard and soft thresholding are often used for denoising signals. For the purpose of compression, the performance of soft thresholding was much lower than hard thresholding. Therefore, we were quickly able to interpret the potential of hard thresholding in the context of compression. Soft thresholding introduces bias during thresolding and shrinks the performance ratio below one, yielding smoother estimates of CDF. In our study, we do not want smoother estimates, rather absolute estimates, resulting in better compression by hard thresholding.

2. Ordered hashing has a stronger effect than random hashing. The KS and CVM ratio statistics was found to be higher with random hashing than with ordered hashing across all distributions. More notably, scenarios varying N and n_i for any distribution type with random hashing yielded in ratios much higher than with ordered hashing. However, we also found that the ratios were more or less stable with comparatively less difference between each observation for scenarios with ordered hashing than with random hashing, indicating a controlled effect of ordered hashing on the outcome.

3. Mixture of Normals yielded different results. Experiments using Mixture of Nor-

mals resulted differently from the trend observed with all other distributions. The scenarios in Mixture of Normals yielded stable ratios greater than one or much closer to one. Also, unlike our observation with other distributions, scenarios with random hashing did not yield much higher performance ratios than with ordered hashing.

4. Through out the experiments, a few parameters behaved similarly across all distributions and sample simulations. An increase in G and N resulted in a decrease in KS and CVM performance ratios. We observed a mixed trend in the ratios for variations in n_i and c parameter values.

In summary, we have conducted experiments using simulation in different scenarios to evaluate the potential of our wavelet based compression method. The results of scenarios that yielded in KS and CVM performance ratios greater than one indicated that our method was effective in estimating CDF based on compressed coefficients using wavelets. There were a few scenarios which yielded performance ratios below one.

It is important to realize that the behavior of the estimated EDF can be different at jumps. The jumps which are smooth and homogeneous are well localized. However, there may be occurrence of non-homogeneous portions of the estimated function, where the localization of jumps may not be as clean. The error entailed by the estimation of the EDF based on compressed coefficients during recovery may be, hence, reflected as a result of the evaluation measure of the reconstructed EDF using KS and CVM performance ratios. However, based on the results from our experiments, we can claim that the method was effective in successfully reconstructing the CDF estimate using few compressed coefficients. This was evident from the results in the default scenario with fixed parameter values and the MC simulation with few parameter variations.

Conclusion

In this study, we determined the EDF of a non-parametric estimate of a univariate sample based on a limited budget of allocated resources for communication. The strategy uses a grid based mapping of common sample data points available at every node in a distributed paradigm. We use the wavelet based compression strategy for determining the CDF estimate, which outperformed the benchmark EDF while respecting the same communication budget. Our study emphasizes on compressing the number of coefficients produced during wavelet transform by careful preserving of only a few selected coefficients.

As the experiments showed, our method was successfully able to produce a good estimate of CDF encouraging a strategy based on compression using wavelets that ultimately depends on a few parameters for CDF estimation within a set communication budget. The evaluation measure of the reconstructed EDF was based on KS and CVM performance ratios which indicated the relative improvement in performance when using our wavelet based approach. It would be interesting to explore methods which are able to perform equally good, further improving upon our results.

In this thesis, we conducted a comprehensive Monte Carlo simulation by varying parameters to observe the effect on the outcome statistics to check effectiveness of our method in respect to the variation in parameter values. Our work showed promising results confirming the usefulness of the wavelet based compression method in CDF estimation. There is a range of parameter values which showed our method worked better. The scenarios which yielded marginal improvement could be explored with other methods which may yield better results. With a better compression strategy, one could also hope to identify the scope of values for which the parameters work best. Another study could also check to find if sensibility to the common sample could have any effect on the results and observations in the study.

We hope that future studies use our proposed method as a building block for developing algorithms for statistical inferences using goodness-of-fit tests and bootstrapping. With growing needs for faster computation in Big Data technologies and remarkable applicability of wavelets for compression purposes, there is a broad scope of exploration and development of wavelet based methods for statistical inference.

Bibliography

[1] F. Abramovich, T. C. Bailey, and T. Sapatinas. Wavelet analysis and its statistical applications. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 49(1):1–29, 2000.

[2] L. Andersson, N. Hall, B. Jawerth, and G. Peters. Wavelets on closed subsets of the real line. *Recent advances in wavelet analysis*, 3:1–61, 1993.

[3] A. Antoniadis. Wavelets in statistics: a review. *Journal of the Italian Statistical Society*, 6(2):97–130, 1997.

[4] L. Baringhaus and N. Henze. A goodness of fit test for the poisson distribution based on the empirical generating function. *Statistics & Probability Letters*, 13(4):269–274, 1992.

[5] R. E. Barlow and H. D. Brunk. The isotonic regression problem and its dual. *Journal of the American Statistical Association*, 67(337):140–147, 1972.

[6] D. Caragea, A. Silvescu, and V. Honavar. A framework for learning from distributed data using sufficient statistics and its application to learning decision trees. *International Journal of Hybrid Intelligent Systems*, 1(1-2):80–89, 2004.

[7] X. Chen and M.-g. Xie. A split-and-conquer approach for analysis of extraordinarily large data. *Statistica Sinica*, pages 1655–1684, 2014.

[8] C.-T. Chu, S. Kim, Y.-A. Lin, Y. Yu, G. Bradski, K. Olukotun, and A. Ng. Map-reduce for machine learning on multicore. *Advances in neural information processing systems*, 19, 2007.

[9] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, jan 2008.

[10] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[11] R. A. DeVore, B. D. Jawerth, and B. J. Lucier. Data compression using wavelets: Errors, smoothness, and quantization. In *Data compression conference*, pages 186–195, 1991.

[12] D. L. Donoho and I. M. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the american statistical association*, 90(432):1200–1224, 1995.

[13] D. L. Donoho and J. M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *biometrika*, 81(3):425–455, 1994.

[14] A. Graps. An introduction to wavelets. *IEEE computational science and engineering*, 2(2):50–61, 1995.

[15] A. HAAR. De cultuur in de in 1906 in zeeland ondergelopen polders. *Versl. en Med. Dir*, 500:86–117, 1910.

[16] V. Honavar, L. Miller, and J. Wong. Distributed knowledge networks. *In Information Technology Conference, IEEE*, 10 1998.

[17] V. Honavar, S. Wong, and A. Mikler. An object oriented approach to simulating large communication networks. *Journal of Systems and Software*, 40(2):151–164, 1998.

[18] Y. Hu, H. Chen, J.-g. Lou, and J. Li. Distributed density estimation using non-parametric statistics. In *27th International Conference on Distributed Computing Systems (ICDCS'07)*, pages 28–28. IEEE, 2007.

[19] M. I. Jordan. On statistics, computation and scalability. *Bernoulli*, 19(4):1378–1390, 2013.

[20] A. Kleiner, A. Talwalkar, P. Sarkar, and M. I. Jordan. A scalable bootstrap for massive data. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 76(4):795–816, 2014.

[21] N. Lin and R. Xi. Aggregated estimating equation estimation. *Statistics and its Interface*, 4(1):73–83, 2011.

[22] P. Ma, M. Mahoney, and B. Yu. A statistical perspective on algorithmic leveraging. In *International Conference on Machine Learning*, pages 91–99. PMLR, 2015.

[23] S. G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE transactions on pattern analysis and machine intelligence*, 11(7):674–693, 1989.

[24] P. Müller and B. Vidakovic. *Bayesian inference in wavelet-based models*, volume 141. Springer Science & Business Media, 2012.

[25] J.-F. Plante. Nonparametric adaptive likelihood weights. *Canadian Journal of Statistics*, 36(3):443–461, 2008.

[26] C. A. Salma, B. Tekinerdogan, and I. N. Athanasiadis. Domain-driven design of big data systems based on a reference architecture. In *Software Architecture for Big Data and the Cloud*, pages 49–68. Elsevier, 2017.

[27] N. Shakhovska, N. Boyko, Y. Zasoba, and E. Benova. Big data processing technologies in distributed information systems. *Procedia Computer Science*, 160:561–566, 2019.

[28] X. Shen and A. Choudhary. A high-performance distributed parallel file system for data-intensive computations. *Journal of Parallel and Distributed Computing*, 64(10):1157–1167, 2004.

[29] X. Shen, A. Choudhary, C. Matarazzo, and P. Sinha. A distributed multi-storage resource architecture and i/o performance prediction for scientific computing. *Cluster Computing*, 6(3):189–200, 2003.

[30] Y. Sheng. Wavelet transform, chapter 10. *CRC and IEEE Press, Boca Raton*, 1995.

[31] L. Tran. From fourier transforms to wavelet analysis: Mathematical concepts and examples. 2006.

[32] S. R. Upadhyaya. Parallel approaches to machine learning—a comprehensive survey. *Journal of Parallel and Distributed Computing*, 73(3):284–292, 2013.

[33] C. Wang, M.-H. Chen, E. Schifano, J. Wu, and J. Yan. A survey of statistical methods and computing for big data. *arXiv preprint arXiv:1502.07989*, 2015.