Process Big Data using Approximation Methods

by

Chen Chen

B.Sc. Dalian University of Technology P.R.C, 2010

M.E. Northeastern University P.R.C, 2012

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

IN THE SCHOOL

OF

Computer Science and Engineering

UNSW
AUSTRALIA

Friday 25th November, 2016

Copyright Statement

Signed .....

Date .....

Authenticity Statement

Signed .....

Date .....

Originality Statement

'I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.'

Signed .....

Date .....

Abstract

Data proliferation makes big data analysis a challenging task. One way to address the issue is to utilize the parallel systems but it is cost consuming. Meanwhile, it has been shown that approximation method with probabilistic guarantee is sufficient in many real applications. It is lightweight, fast and quality-guaranteed. In this thesis, we apply sampling and sketching techniques to obtain high-quality approximation results for four important queries.

Firstly, we study the gapped set intersection size estimation problem. Given two integer sets and a gap parameter δ, two elements are deemed as a match if their numeric difference equals δ or is within δ. We first distinguish two subtypes of the estimation problem: the point gap estimation and range gap estimation. Then we propose optimized sketch to tackle the two problems efficiently and effectively with theoretical guarantees. We also demonstrate the usage of our proposed techniques in mining top-K related keywords efficiently.

Secondly, in response to the emerging call to a fast data visualization, we study the order preserving estimation problem that can retain important data characteristics. We focus on the population mean as our primary estimation function. By dynamically allocating the failure probability, we propose two effective query processing strategies that can preserve the estimated order to be correct with probabilistic guarantees.

Finally, motivated by the demand of analyzing large graphs, we investigate two related concepts in social networks, which are influence maximization and closeness centrality. In order to accelerate the process of influence maximization problem, we bring the order of samples into the RIS framework and derive early stop conditions to accelerate the seed set selection procedure. Furthermore, we provide a cost-effective method to find a proper sample size to bound the quality of the returned seed set. For closeness centrality, we extend the concept to a set of nodes. We aim to find a set of k nodes that has the largest closeness centrality as a whole. We show that the problem is NP-hard, and prove that the objective function is monotonic and submodular. Hence a greedy algorithm can return a result with $1-1/e$ approximation ratio. In order to handle large graphs, we propose a sampling based approach. We reduce the cost of computing shortest path distances from the sampled nodes to the other nodes by selecting the nodes incrementally. In addition, optimization techniques are developed to further reduce the cost of updating nodes' closeness centralities.

Publications

- Chen Chen, Jianbin Qin, Wei Wang. On Gapped Set Intersection Size Estimation. CIKM 2015. (Chapter 3)
- Chen Chen, Wei Wang, Xiaoyang Wang. Effective Order Preserving Estimation Method. ADC 2016. (Chapter 4)
- Xiaoyang Wang, Ying Zhang, Wenjie Zhang, Xuemin Lin, Chen Chen. Bring Order into the Samples: A Novel Scalable Method for Influence Maximization. TKDE 2016. (Chapter 5)
- Chen Chen, Wei Wang, Xiaoyang Wang. Efficient Maximum Closeness Centrality Group Identification. ADC 2016. (Chapter 6)
- Xiaoyang Wang, Chen Chen, Ying Zhang. EDMS: A System for Efficient Processing Distance-Aware Influence Maximization. ADC 2016.

Acknowledgements

First of all, I would like to deliver my great gratitude to my supervisor Prof. Wei Wang for his support and guidance. I thank him for guiding me constantly through the road of my research and encouraging me to overcome the difficult times. Also, I learnt the characteristics of an excellent researcher - passion, diligence and earnestness in the research.

Secondly, I would like to express my sincere gratitude to my co-supervisor, Prof. Xuemin Lin for his encouragement. Without his consistent and illuminating instruction, this thesis could not have reached its present form.

Thirdly, parts of the work in this thesis were conducted in collaboration with Dr. Jianbin Qin, Dr. Xiaoyang Wang, Dr. Ying Zhang and Dr. Wenjie Zhang. I thank them for supporting the work presented in this thesis.

Last but not least, I am very thankful to my beloved family and friends for their love, support and encouragement.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The ever-increasing amount of data is prevalent and arises in many forms, for example, activity data from GPS locations and social network activities, bioinformatics data from genetic sequences and business data from customer behavior trackings. Consequently, data proliferation makes big data analysis a challenging task. One way to address the issue is to develop massive storage architectures and parallel systems such as MapReduce to compute the results. However, this approach may be cost consuming. Meanwhile, it has been shown that approximation method with probabilistic guarantees is sufficient in many real applications. It is lightweight, fast and quality-guaranteed. In this thesis, we apply sampling and sketching techniques to obtain high-quality approximation results on various types of queries. The advantage of data synopses is that they can use very little space and supply fast approximate answers to the queries. Therefore, we can achieve a balance among result quality, data characteristics and resource constraints.

The thesis studies how to design lightweight and fast approximation algorithms for large datasets over four queries. Specifically, for integer sets, we define the gapped set intersection size estimation problem, and demonstrate the usage of

our proposed techniques in mining top-K related keywords efficiently. In order to generate fast data visualization while retaining crucial data characteristics, we study the order preserving estimation of population averages. For graph data, we investigate the problem of fast solving influence maximization problem, and how to efficiently identify a set of nodes which has the largest closeness centrality as a group.

In Section 1.1, we briefly describe the motivations of the problems, the challenges faced and the general ideas of our approaches. Section 1.2 summarizes the contributions of this thesis for each problem studied. Thesis organization is presented in Section 1.3.

1.1 Motivations and Our Approaches

1.1.1 On Gapped Set Intersection Size Estimation

In information retrieval, the search engine needs to intersect the positional inverted lists of query keywords to answer a multiple keyword phrase query. A state-of-the-art query processing method, *svs*, performs binary intersection using a heuristic order that is purely based on the length of the inverted lists [CM10]. This heuristic is not effective when search keywords are not very selective (e.g., "`to be or not to be`"). It is possible that a pair of query keywords with the positional constraint imposed by the phrase query will result in very small intermediate result size (e.g., "`be`" followed immediately by "`or`"). Hence, if we can estimate its cardinality accurately and efficiently, we may find a better inverted list intersection order to process such queries. Motivated by the example, we define and study the problem of gapped set intersection size estimation (abbreviated as GSISE). Therefore, above application is modeled as a gapped set intersection size estimation problem with a

point gap constraint of 1.

Given two sets S_A and S_B, set intersection $S_A \cap S_B$ is to find all the common elements from two sets. A common case is that all elements in the set are integers (e.g., document IDs or positions in an inverted index). Hence, the common element pair (a, b) satisfies $b - a = \delta$, where $a \in S_A$, $b \in S_B$ and $\delta = 0$. We generalize the set intersection on integer sets to allow for "gaps" (i.e., $\delta > 0$). We define two primitives: the point gap constraint corresponds to a fixed gap of δ, and the range gap constraint corresponds to a gap of size no larger than δ.

For the proposed queries, we are interested in methods that can estimate these gapped set intersection size efficiently and accurately. For the estimation problem with point gap constraints, we propose a basic method that reduces the problem to the standard set intersection size estimation problem, which can be solved using the state-of-the-art sketch technique. However, the index space is linear to the maximum query gap allowed. We improve it by judiciously selecting a subset of sketches to construct, and this reduces our sketch size from $O(N)$ to $O(\sqrt{N})$. The space cost of our approach is proved to be asymptotically optimal, while the time complexity remains the same. For the estimation problem with the range gap constraint, a baseline method is to reduce the problem into multiple estimation problems with different point gap constraints, and this requires estimation time linear to the gap size. We propose an extension of the bottom-k sketch to multiset and an unbiased estimator for inner product. We achieve an accurate and fast estimation method that is independent of the gap size. We also demonstrate the usage of our proposed techniques in mining top-K related keywords efficiently by integrating with an inverted index.

1.1.2 Effective Order Preserving Estimation Method

Data visualization is widely adopted to communicate information clearly and effectively. It enables data analysts to visually see the underlying dataset, so they can grasp important trends or identify new patterns. However, the large amount of data in many real applications may consume a very long time to produce such visualizations. We observe that it is possible to use sampling method to solve the problem by capturing vital characteristics of the original data while typically consuming much less resources. One of the most widely used characteristics is the population mean, which is the average value of a group of numeric data. The methods and criteria for a reliable population mean estimation are well developed in both statistical inference and computer science areas. However, when it comes to several groups of data, it has long been neglected that an order estimation on the population means is of equal importance. Motivated by the observation, we study the order preserving estimation (OPE) problem. Given k groups of numeric data with unknown distribution along with $\delta \in (0, 1)$, OPE returns an order estimate on the group average with a probabilistic guarantee $1 - \delta$. We consider the total sample size as the cost function in our algorithm design, as we notice that the offline sample processing time is less concerned with the rapid development of modern computing ability. Instead, the sample size becomes a critical factor to be restrained. For example, the I/O cost of obtaining data from external or distributed storage is very high. Or in most clinical trials of new drugs, the number of human subjects is usually very limited due to the risk or ethical issues. In both cases, it is not affordable to get as many samples as we want.

We use progressive sampling method to tackle the problem. Specifically, we design two stop functions, named Interval Separation and Pairwise Comparison. They utilize the relation among current sample means to make judicious decisions,

in order to output a correct result while using as little samples as possible. We also propose a heuristic sample strategy to assign new sample points adaptively. Specifically, Interval Separation reduces order estimates to separating the underlying confidence intervals. Due to tail inequality, all true means can be bounded within their confidence intervals with high probabilities. Thus as long as the intervals do not overlap, the order can be guaranteed. Next in the Pairwise Comparison method, we first show that minimizing the sample size in OPE is NP-hard by reducing from TSP problem on tournament graphs. In order to compute the failure probability, we introduce the prefix downsample technique to equalize the sample size between adjacent sample groups.

1.1.3 A Novel Scalable Method for Influence Maximization

As a key problem in viral marketing, influence maximization has found many important applications in real life. Given a positive integer k, it aims to find a set of k users in a social network, which can make the largest of adoption or cascade of information. For example, a company would like to promote its new product by choosing some influential users. By offering some incentives, the company expects that the influential users can propagate the product information through their social networks, which leads to a large adoption of the new product finally.

Recently, Borgs *et al.* [BBCL14a] develop an elegant framework, called reverse influence sampling (RIS) to solve the influence maximization problem. [BBCL14a] conducts an in-depth theoretical analysis of the sample size needed to bound the effectiveness. However, due to the large constant factor in the sample size, it does not work well in practice. Tang *et al.* [TXS14] propose TIM that improves the sample complexity and show that the sample size required is at least λ/OPT, where OPT is the influence spread of the optimal seed set, and λ is an equation

related to the error parameters. Later in [TSX15], the authors propose IMM, which is the state-of-the-art method. It utilizes the martingale technique to further reduce the sample size and reuse the samples. IMM shows that the sample size required is λ^*/OPT, where λ^* is smaller than λ.

Although IMM offers good performance in solving the influence maximization problem, it still remains much room for improvement in term of efficiency and scalability. IMM needs to get a tight lower bound of OPT to determine a tight sample size which causes a heavy overhead. Because it needs to iteratively double the sample size and select k nodes in the current samples to refine the lower bound of OPT. The cost cannot be neglected when k and n are large. In a nutshell, three approaches [BBCL14a, TXS14, TSX15] based on RIS can be implemented in a two-phase framework: 1) determine a sample size, 2) select k nodes based on the samples. Approach in [BBCL14a] is inefficient due to the large sample size, while TIM and IMM are limited by the heavy overhead in the first phase. Consequently, instead of trying to obtain a tight sample size, we aim to accelerate the second phase over a reasonable large sample size.

We consider to bring the order of samples into the RIS framework to improve the performance. We propose a bottom-k sketch based RIS framework, which can achieve possible early termination without enumerating all the samples. However, in the worst-case scenario, if there are not enough (*i.e.,* less than k) nodes that can meet the framework's requirement, we still need to materialize all the samples. Thus in order to guarantee the efficiency and the result quality, we develop a quick sample size estimation approach based on the small world property. Specifically, we provide a cost-effective method that efficiently obtains a lower bound of OPT. By feeding our lower bound into the sample size equation in IMM (*i.e.,* λ^*/OPT), we can obtain a sufficient sample size required. Also, we develop several optimizations

to accelerate the generation of sample order and the processing of the worst-case scenario.

1.1.4 Maximum Closeness Centrality Group Identification

As a subject of broad and current interest, social networks have been widely studied for decades. A social network is usually represented as a graph $G = (V, E)$ where V denotes the set of nodes and E denotes the set of edges. Centrality, which measures the importance of a node in a social network, has been a fundamental concept investigated in the social networks. There are different measurements of centrality developed for various purposes, such as closeness centrality [Bav50], betweenness centrality [AGW15], eigenvector centrality [BL15a], etc. We focus on the classic closeness centrality, which is defined as the inverse of the average distance from a node to all the other nodes in the social network. The distance between two nodes is calculated by the shortest path distance. The smaller the average distance of a node is, the more important or more influential the node will be. To find the influential nodes (users) in a social network, many research efforts have been made to find the k nodes with the largest closeness centrality [EW01, OCL08, OLH14]. However, in many real applications, such as team formation, we may need to find a set of k users which has large closeness centrality as a *group*, instead of returning the k independent users in the top-k ranking. Such observation motivates us to extend the definition of closeness centrality for a single node to a set of nodes as a whole. Specifically, the closeness centrality of a set S of nodes is defined as the inverse of the average distance from S to the nodes in G. And the distance from S to a node $u \in V$ is defined as the minimum distance from u to the nodes in S. The maximum closeness centrality group identification problem is to find a set of k nodes in the social network with the largest closeness centrality.

The challenges of the problem lie in two aspects. First, we show that the problem is NP-Hard. Fortunately, we prove that the objective function is monotonic and submodular. It means we can obtain a result with $1-1/e$ approximation ratio by adopting a greedy framework. Second is that we still need the information of the all pairs shortest path distances even for the simple greedy algorithm, which is prohibitive to compute ($O(|V|^3)$ time) and to store ($O(|V|^2)$ space) when the graph is large. In order to scale to large graphs, we propose a sampling based approach by extending the traditional sampling method for estimating the closeness centrality of a single node. In addition, we develop a strategy to identify the nodes incrementally. Then we utilize the selected nodes to reduce the cost of computing the distances from the nodes to the samples. To further accelerate the process, we develop optimization techniques to reduce the update cost for the less important nodes.

1.2 Contributions

We propose efficient techniques to deal with four important problems by utilizing sampling and sketching techniques. For each of these queries, we briefly describe our contributions.

On Gapped Set Intersection Size Estimation.

- To the best of our knowledge, this is the first work to formally define the point and range gapped set intersection size estimation problems, which can be used as primitive operations in a wide spectrum of applications.
- We design space and time efficient sketch and estimation methods for both types of estimation tasks. Our estimates are unbiased and have theoretical guarantees.

- We demonstrate the application of our technique for approximately mining top-K related keywords from large document collections. Our technique is especially useful in this application scenario where an exact solution requires orders of magnitude larger space and time.
- Comprehensive experiments on the ClueWed09 dataset demonstrate the efficiency and accuracy of the proposed methods.

Effective Order Preserving Estimation Method.

- We design the Interval Separation stop function by reducing the order estimates to separating the confidence intervals dynamically.
- We introduce the prefix downsample technique in the Pairwise Comparison stop function, and prove that minimizing the sample size in OPE is NP-hard.
- In order to output a correct result while using as little samples as possible, we propose a heuristic sample strategy to assign new sample points adaptively.
- We conduct empirical evaluations on both synthetic and real datasets to demonstrate the effectiveness of our proposed methods.

A Novel Scalable Method for Influence Maximization.

- We propose the BKRIS framework, which accelerates the RIS framework by involving the order of samples based on bottom-k sketch.
- We propose an efficient method to derive a sufficient and reasonable large sample size by using the small world property.
- We provide novel techniques to optimize the generation of sample order and to efficiently handle the worst case.

- We experimentally evaluate BKRIS on 10 datasets, and show that we can achieve up to 2 orders of magnitude speedup compared with the state-of-the-art approach IMM under both IC model and LT model.

Maximum Closeness Centrality Group Identification.

- We propose a sampling approach by extending the traditional sampling method for the single node closeness centrality estimation.
- We develop an algorithm to incrementally select nodes, in order to reduce the cost of computing the distances from the nodes to the samples.
- We develop optimization techniques to reduce the update cost for the less important nodes.
- Through experiments on real world social networks, we verify the efficiency and effectiveness of the proposed techniques.

1.3 Organization

This thesis is organized as follows.

- Chapter 2 provides a survey of the related work.
- Chapter 3 describes our techniques for estimating gapped set intersection size and its application.
- Chapter 4 presents our algorithms for effective order preserving estimation.
- Chapter 5 presents our novel scalable methods for the influence maximization problem.

- Chapter 6 presents our approaches for efficient maximum closeness centrality group identification.
- Chapter 7 concludes our research and provides several possible directions for future work.

Chapter 2

Related Work

In Section 2.1, we introduce the related work of approximation methods for set queries. In Section 2.2, we describe the existing techniques about influence maximization problem. Finally, we present the related techniques of closeness centrality in Section 2.3.

2.1 Approximation Methods for Set Queries

In this section, we first introduce the basic sampling schemes and accuracy measurements in 2.1.1, then survey several key approximation methods that are applicable to set queries in 2.1.2.

2.1.1 Basic Sample Schemes and Concentration Inequalites

We categorize some basic sample schemes from high-level statistical aspects. We also describe the variables for quantifying the estimation accuracy and the concentration inequalities for measuring the error bound.

Random sample with replacement and random sample without replacement are two sampling schemes of simple random sample. Conceptually, simple random

sample is a basic type of sampling, as it usually serves as a component of other more complex sampling methods.

Random sample with replacement. Given a finite population of size N, each data point can be identified by a unique key. Let $U = \{1, 2, \cdots, N\}$ be the set of these keys. Then in order to extract a sample of fixed size n logically, we can repeat the following steps n times.

- Generate a random number $i \in [1, N]$ uniformly.
- Retrieve the i-th point from the data then add it to the current sample set.

The obtained sample can be denoted by a vector $(s_1, s_2, \cdots, s_N) \in \mathbb{N}^N$, where s_i is the number of times that point i is included in the sample, and $\mathbb{N} = \{0, 1, 2, \cdots\}$. Besides, the sample size n can be greater than the population size N. A basic query type associated with random sample with replacement is the population average `AVG`. Denote by X_i the value of point i, the estimator for `AVG` is $\frac{1}{n}\sum_i X_i s_i$, which is simply the average of the sampled values. It is an unbiased estimator as the expectation of the estimator is equal to the statistics to be estimated. In terms of sampling process, we can consider it as performing a sequence of independent trials over the random variables. A property of the random variables is called independent identically distributed (i.i.d.). It is an important assumption in the statistical analysis as it can facilitate the derivations hence enables us to focus on the essential information from the underlying data. The advantage of with-replacement sample is that it is the only type of sampling where a sample of size n can be viewed as a sequence of trials on n i.i.d. random variables. As will be shown later in this section, much of classical statistical analysis such as central limit theorem or concentration inequalities will apply straightforwardly to such a case. However, a disadvantage of random sample with replacement is that it may

cause higher variance than the without-replacement scheme when given the same sample size.

Random Sample without Replacement. A major difference between with-replacement and without-replacement sampling is that the latter only allows each point to appear once in the sample. To extract a sample of fixed size n logically, we can repeat the following steps n times.

- Generate a random number $i \in [1, N]$ uniformly.
- Retrieve the i-th point from the data then add it to the current sample set, if it has not been added before.

The obtained sample can be denoted by a vector $(s_1, s_2, \cdots, s_N) \in \{0, 1\}^N$, where s_i is an indicator variable with value 1 if point i is included in the sample, otherwise s_i is defined as 0. Under the process, the largest sample size will be the size of population. In such case, we obtain the entire population as a sample thus can answer queries accurately. Note that in the with-replacement sample scheme, the variance can still be greater than 0 when the sample size is N. The reason is that it allows a point to appear more than once in the sample. Intuitively, it results in less information included in the sample of a fixed size, which explains why without-replacement scheme usually achieves lower variance.

Stratified Sample. To further reduce the variance in the without-replacement sampling, we can use the stratified sampling method. The idea is to divide the dataset into several strata, and perform random sampling within each stratum independently. It provides a natural way to reduce the variance for simple random sample, as we can construct strata with internally well-concentrated values even if the data set tends to be heterogeneous. In the database literature, stratification is applied in [HS92] to increase the accuracy of join size estimation. [CDN07] uses

query workload information to divide the data into strata in order to maximize the query accuracy. From the statistical point of view, one of the classic results in the stratified sampling theory is the Neyman allocation [Ney34]. Given a fixed total sample size, it allocates sample size for each stratum in order to minimize the total variance.

Recently, the authors in [KBP+15] design a visualization system that preserves the visual property of ordering population means. The authors improve the round-robin stratified sampling method, and prove the near optimality in the total sample size. However, an important assumption for the near optimality is that it does not consider the inactive intervals to be back to active. The correctness of the order estimation is based on the reasonable estimates at each sample round. Specifically, for a newly added sample point, the updated confidence interval contains the true mean with probability more than $1 - \delta/k$, where k denotes the total number of groups. As all true means are bounded by their confidence intervals, ordering them correctly reduces to separating the underlying intervals. The merit is its simplicity for understanding. Nevertheless, it does not seem to be tight enough empirically.

For the accuracy measurements, bias and variance are essential statistics. In the study of statistical inference, bias and variance can provide instructive information on the quality of an estimator [CB02]. In the machine learning literature, the bias-variance tradeoff is used to analyze a learning algorithm's expected generalizaiton error [AMMIL12]. However, in many real applications, the users may prefer answers described by a confidence interval [Ney37] with probabilistic guarantees. Given parameters ϵ and δ specified by the user, a confidence interval can be interpreted as "the probability of true answer deviating from the estimate is upper bounded by δ". There are numerous ways to construct a confidence interval, such as central limit theorem and its variants, and different forms of concentration inequalities.

In this thesis, we use concentration inequalities as our primary tool to analyze the quality of an estimation as it generally does not have assumptions on the underlying data distribution.

Concentration Inequalities. Basically, concentration inequalities are designed to give a sharp prediction of the query by bounding the estimation errors with a given probability. The requirements of higher-order moments will lead to sharper bounds on the concentration probabilities. Concrete examples are from Markov inequality with first moment existence to Hoeffding bound with the existence of the moment generating function. Markov's inequality applies when a random variable is non-negative and has bounded expectation. If the random variable also has a finite variance, Chebyshev's inequality can be adopted. But none of the them can give us an exponentially decreasing bound as Hoeffding bound [Hoe62]. The Hoeffding bound provides a sharper bound, yet it requires the independence assumption. Historically, it is closely related to Chernoff bound [Che52], Bersntein inequality, and McDiarmid's inequality [McD89], etc. For the case where the independence does not hold, Serfling considers the impact of the sample ratio and introduces a general concentration inequality for sampling without replacement in [Ser74]. As the sample size approaches the population size, Serfling's bound leads to large improvement, which coincides with the intuition for our without-replacement sample analysis discussed previously. Recently, Bardenet *et al.* [BM15] propose an empirical Bernstein-Serfling bound for practical applications which works with sampling without replacement with an unknown variance.

2.1.2 Hash based Techniques for Set Queries

Set query is a fundamental problem in many areas. Due to its importance, the problem has been investigated for quite a while in the field of databases. Among

various query types, we mainly focus on estimation methods for intersection size and Jaccard similarity in this thesis. There are plenty of literature focusing on exact set intersection problem [DLM00, DK11]. The algorithms working on sorted set are mostly used [SWL11]. By applying linear merge of two sorted set S_A and S_B, the set intersection can be finished in $O(|S_A| + |S_B|)$ time by iterating over each element of the sorted sets. However, when the set sizes are unbalanced, the method is inefficient. In [HL72], authors propose a set intersection approach requiring $O(|S_B| + \log \binom{|S_A| + |S_B|}{|S_A|})$ when $|S_A| \ll |S_B|$. There are also work further improving the performance by utilizing hashing and adaptive methods [BLLS09, BLL06].

Hash-based algorithms are widely adopted to estimate the intersection size and Jaccard similarity [Coh97, BCFM98, BHR+07, CK07, PSW14]. They utilize various hash functions to keep repeated items in order to achieve consistency over different or distributed observations. First proposed in [Coh97] in the form of k-mins, the technique is used to estimate the size of reachability set and the transitive closure. Meanwhile, Broder *et al.* [Bro97, BCFM98] independently invent MinHash technique for Jaccard similarity estimation. Then the authors in [CK07, CK08, CK09] further refine the k-mins technique and propose the bottom-k sketch. Bottom-k can be categorized as a kind of hash-based sampling method, and is very effective in answering set queries. There are several methods with different names that share the same concept, *e.g.,* MinHash [BCFM98] and KMV [BHR+07].

In the derivation of bottom-k and many sum-based estimation, Horvitz-Thompson (HT) estimator [DGH52] is a key component. The estimation is an unbiased sampling based method. After specifying the predicate, HT estimator essentially returns a sample sum, each of which is processed using the sampled values satisfying the predicate and the corresponding inclusion probabilities. The

inclusion probability refers to the probability of a data point being included in the sample. The idea is similar to the intuition of the indicator variables. Its advantage is that we do not need to know queries when sampling. However, there are cases where the inclusion probability is unknown during the estimation phase, thus the HT estimator cannot be obtained directly. [CK08] uses conditional probability to calculate the inclusion probability. Combined with weighted sampling without replacement, it is able to get the rank-conditioning adjusted weight. Another case that the HT estimator would fail is when the predicate cannot be expressed in the form of function of sums, such as `MAX` and `MIN` queries. Estimating extreme values is a difficult task as the dataset may have arbitrary data distributions. It is generally impossible to design an effective estimator without knowing some domain knowledge. Authors in [WJ09] propose sampling based methods for estimating extreme values by learning certain prior information under the Bayesian framework. They use previously observed queries to make a guess about the type of current query.

There are also papers focusing on Jaccard similarity between multisets. A simple extension of similarity measure from set to multiset has been shown in [HGI00]. Given a multiset, they consider creating a new set with distinct elements for each copy of a given element (face value) in the multiset. Specifically, if f_x is the number of occurrences of face value x in the multiset, we will replace x by the pairs $\{(x, i) \mid 1 \leq i \leq f_x\}$. After the transition, MinHash on set can be generalized to the multiset above. However, the time complexity of generating such a hash function grows linearly in the total weights of face values. In addition, the method would fail when the weight is required to be real values.

To deal with the problem of efficiency, authors in [GP06] propose a method with a $\log f_x$ running time. Instead of generating all the hash values of each copy, they adopt the idea of inverse sampling to skip certain copies. The specific quantile

function they use is derived from geometric distribution, as the distribution can simulate the waiting time of occurrence of an event. The intuition is that certain hash values happen to be the waiting time that we would like to skip. What we really need is the hash values that interleave between the waiting times. After generating the smallest hash value of each face value, the algorithm returns the minimum one along with the corresponding active index as the MinHash signature for the multiset.

Later in [MMT07] the authors solve the case of real value weights and give an algorithm to return consistent weighted MinHash in expected constant time. They describe two properties that a correct method should possess in order to return a right Jaccard similarity between weighted set, uniformity and consistency, respectively. Uniformity comes in two aspects. Given a face value x and its weight f_x, each copy of x should have equal probability of being sampled as a representative. The second aspect is throughout the weighted set therein. The probability of each representative being sampled as the weighted MinHash is proportional to its weight. Consistency is very important in reducing the estimation variance. It guarantees that similar items can be sampled as signatures simultaneously hence a hash collision can be detected.

Ioffe [Iof10] further improves the complexity of [MMT07] to a worst case constant time. The method avoids to generate the sequence of active index, but instead to study their distribution. A sequence can be uniquely determined given the reccurence relation of the sequence. Under such strategy, the method is able to generate only two random variables to extract the active index in each bucket, hence reduces the time complexity to the worst-case constant time.

For the standard MinHash technique, the number of random permutations $k = O(\frac{1}{\sqrt{\epsilon}})$, where ϵ is the expected relative error. Thus more permutations will reduce

the variance of the estimator. Based on this observation, Li *et al.* [LK10] introduce the idea of building meta-sketch based on the standard MinHash, in order to save space cost per single permutation. Instead of keeping all 64 (or 32) bits of each hashed value, the algorighm stores the lowest b bits. Since when the similarity is high, few bits of each hashed value are enough to determine the resemblance. In other words, there exists information redundancy in hashed value to be compared. The method of [LK10] is to trade redundancy for permutations when space is limited. Mitzenmacher *et al.* [MPP14] deal with the redundancy in another way. Rather than crudely keeping the lowest b bits for each hashed value, the authors apply a random hash function on top of the standard MinHash, and obtain a bit vector of length n. Each position of the bit vector records the parity of the number of items that have been hashed into the position. The technique can be considered as a special case of the bloom filter. It uses one hash function with the odd feature where the usual OR is replaced by XOR. By using such hierarchical structure, the information in the original MinHash is retained as a whole. Although the methods in [LK10] and [MPP14] can achieve small variances while keeping a constant space cost, there remain several issues. Both methods would fail if the similarity is low and the techniques in [MPP14] can only deal with a similarity estimation between two sets. To deal with a low Jaccard similarity, ATLAS [ZLG11] modifies the standard "intersection then union" composition of MinHash and uses them inversely as "union then intersection" to generate the candidate set.

2.2 Influence Maximization

In this Section, we review the related work of influence maximization. In Section 2.2.1, we present the existing techniques for answering influence maximization

problem. In Section 2.2.2, we describe the methods related to query-based influence maximization, in which the selected seed sets will be different when the input query changes.

2.2.1 Influence Maximization

As a key problem in viral marketing, influence maximization problem has been widely studied in the literature [DR01, RD02, CWW10, CYZ10, GLL11b, BBCL14a, TSX15, TXS14, CDPW14b, BL15b]. Given a social network G and a positive integer k, influence maximization aims to find a set of k nodes, which can maximize the expected influence spread under a certain diffusion model $\mathcal{M}$. Since the information is propagated through close friends, families and co-workers, etc., the promotion strategy is shown to be more effective than the traditional media channels, such as TV and newspaper advertisements [CWY09].

The influence maximization problem is formally defined in [KKT03]. The paper introduces two diffusion models to describe the information propagation, which are independent cascade model and linear threshold model. The two models emphasize different features for information propagation and are widely adopted. The independent cascade model emphasizes that the influence between friends are independent. It means the probability of u_1 influencing u_2 does not depend on u_1 and u_2's friends. While the linear threshold model works in a different manner, it emphasizes that for a user u, if most of his friends are influenced by certain ideas (*e.g.,* watching NBA), u will adopt this idea as well. The details of the two models are presented in the following.

Independent Cascade Model. The input of independent cascade model is a directed graph $G = (V, E)$, where each edge $e_{u,v}$ is associated with a probability $\mathbf{Pr}[u, v]$ within $[0, 1]$. It denotes the influence from u to v, *i.e.,* each user u can

influence its out-going neighbour v with probability $\mathbf{Pr}[u, v]$. Given a set S of nodes, let S_i denote the set of nodes activated at time i. The influence propagation works as follows.

- At time 0. Only nodes in $S_0 = S$ are *active*, while the other nodes are *inactive*.
- At time i. Each node u in S_{i-1} will try to influence each inactive neighbour v with probability $\mathbf{Pr}[u, v]$. If a node is activated, it will stay active for the subsequent iterations.
- The propagation repeats above steps until there is no node can be activated, *i.e.,* $S_t = \emptyset$, where $t = 0, 1, 2, \cdots$.

Linear Threshold Model. The input to the linear threshold model is also a directed graph $G = (V, E)$. Differently, each edge $e_{u,v}$ is associated with a weight $w(u, v)$, and $w : V \times V \rightarrow [0, 1]$. If there is no edge between two nodes, the weight is set to 0. For a node $u \in V$, let $N_{in}(u)$ be the set of u's in neighbours. Then for each node $u \in V$, the weights of edges should fulfill the following equation.

$$\sum_{v \in N_{in}(u)} w(v, u) \leq 1.$$

Let S_i denote the set of nodes activated at time i, then the influence propagates for a set S of nodes work as follows in the linear threshold model.

- At time 0. Only nodes in $S_0 = S$ are *active*, while the other nodes are *inactive*. For each node u, we generate a random number t_u from [0,1], denoting the probability of u being influenced.
- At time i. If the sum of its active neighbours' edge weights is greater than

t_u, *i.e.,*

$$\sum_{v \in \bigcup_{0 \leq j \leq i-1} S_{i-1} \cap N_{in}(u)} w(v, u) \geq t_u,$$

then the node is activated and it will stay active for the subsequent iterations.

- The process terminates if there is no node can be further activated.

Finally, for both diffusion models, *i.e.,* the independent cascade model and the linear threshold model, the influence spread is calculated as the expected number of nodes that can be influenced by using the above procedure, *i.e.,* $\mathbb{E}[\sum_{i \in [0,t]} S_i]$.

In the seminal paper [KKT03], the authors study the influence maximization problem by adopting the two models. In addition, the authors show that both diffusion models can be generalized and they are equivalent after the generalization. Under both diffusion models, the influence maximization problem of finding k nodes is NP-hard. Fortunately, the objective functions are monotonic and submodular. Kempe *et al.* [KKT03] propose a greedy algorithm that iteratively selects the node with the largest marginal influence. The algorithm is simple yet can return a result with $1 - 1/e - \epsilon$ approximation ratio. The ϵ error is involved as it uses the Monte Carlo simulations to compute the influence spread, which is proved to be a #P-Hard problem under both models [CWW10, CYZ10].

However, the performance of the greedy algorithm is not satisfactory, because it needs to run a large number of Monte Carlo simulations. Due to the importance of the problem, there is numerous follow-up work that aim to improve the efficiency of the naïve greedy method. A major limitation in the naïve greedy algorithm is that it needs to update the influence spread for each node in each iteration. While for most of the nodes, their influences are quite small hence they would not be selected into the final seed set. If we are able to reduce the number of influence calculations, the efficiency could be greatly improved. In [LKG+07], the authors

propose a lazy-forward method called CELF, which is based on the idea that if a node u's marginal influence is greater than v's influence, then v can be pruned from the current iteration. Thus the method can calculate the influence spread only when necessary. According to the experiment results, the proposed method achieves 700 times speedup compared with the naïve method. [GLL11b] proposes the CELF++ method, which further accelerates the query processing by pruning the insignificant nodes. It reduces the cost of CELF by $35\% - 55\%$ according to the experiment evaluations.

Even though CELF++ significantly reduces the cost, it still cannot process large graphs efficiently. Hence, some heuristic methods are developed to improve the performance. In [KS06], a shortest path based heuristic is provided, as well as an efficient update method. In [CWY09], Chen *et al.* propose an algorithm to reduce the computation for the case where each edge has identical probability. In addition, the authors propose a heuristic degree discount based method to efficiently return a set of nodes. In [CWW10], the authors propose a tree-based heuristic for the independent cascade model, in which the influence propagation between two nodes only goes through the path with the largest probability. Based on this heuristic, it develops the MIIA and MIOA structures for each node u, which are two tree structures rooted at the node u. MIIA(u) assembles all the nodes that can influence u, while MIOA(u) assembles all the nodes that can be influenced by u. Through MIOA(u), we can approximate the influence spread of u in polynomial time. In addition, efficient update algorithm is proposed to find a set of k nodes. By using the similar idea, [CYZ10] applies the tree based structures to solve the influence maximization problem under the linear threshold model.

Unfortunately, as the size of social networks grows, these heuristic methods are still not able to process very large graphs. In addition, due to the approximation

of the influence calculation, they cannot offer theoretical guarantees held by the naïve greedy algorithms. To bridge the gap between efficiency and effectiveness, Borgs *et al.* [BBCL14a] propose a new sampling framework, *i.e.,* reverse influence sampling (RIS). Based on the two probabilistic influence propagation models, a social network G can be considered as a set of graph distributions, where each instance is generated following the propagation procedure. For example, in the independent cascade model, each instance g is generated by removing each edge with probability $1 - \mathbf{Pr}[u, v]$. The idea of the RIS model is that given a set of randomly sampled instance-node pairs, if a set S can reach the sampled node frequently, it means S will have larger influence. Given a set of randomly sampled instance-node pairs with size θ, the authors show that Equation 2.1 is an unbiased estimation of influence spread.

$$\hat{I}(S) = n \times \frac{F(S)}{\theta}, \tag{2.1}$$

where $n = |V|$ is the total number of nodes in G, $\hat{I}(S)$ is an unbiased estimation of S influence spread, and $F(S)$ denotes the number of sampled nodes that can be reached by S. Given the estimator and a set of samples, we can greedily select the node with the largest marginal coverage. Then the method can return a result with $1 - 1/e - \epsilon$ approximation ratio, where ϵ is decided by the sample size. Since the computation time is determined by the sample size, the main problem is to derive a proper sample size. In [BBCL14a], the authors only consider the theoretical analysis where the sample required is computed as the number of edges visited. It renders a large constant factor in the sample size equation. [TXS14] reduces the sample size of the RIS model and makes it work in practice. Tang *et al.* [TSX15] further improve the performance of [TXS14] by utilizing the martingale technique hence manage to reuse the samples. As shown in [TSX15], if the sample size satisfies Equation 2.2, it can return a result with $1 - 1/e - \epsilon$ approximation ratio and the

success probability is at least $1 - n^{-l}$.

$$\theta = \frac{2n((1-1/e)\cdot\alpha+\beta)^2}{OPT\cdot\epsilon^2}, \tag{2.2}$$

where OPT is the influence spread of the optimal seed set. α and β are defined as follows:

$$\alpha = \sqrt{l\log n + \log 2}$$
$$\beta = \sqrt{(1-1/e)(\log\binom{n}{k} + l\log n + \log 2)}.$$

2.2.2 Query-based Influence Maximization

For the general influence maximization problem in the previous section, the returned seed sets will be identical when the input k is given. It means that it is not target specified. However, for different purposes of promotion in real applications, the selected seed set should be different. To meet this requirement, different variances of influence maximization problems are proposed. Apart from the input social network and the selected seed set size k, users are allowed to input other parameters. Based on different parameters (*i.e.,* purpose of promotion), the final returned seed set will be different. We denote these variances as query-based influence maximization. In this section, we review two major types of variances, *i.e.,* topic-aware influence maximization and location-aware influence maximization.

Topic-Aware Influence Maximization. In topic-aware influence maximization model, the topic information is taken into consideration when computing the propagation probability of each edge. The topic-aware influence maximization problem is introduced in [BBM12], where the probability over each edge varies when the input topics are different. Specifically, each edge is associated with a vector $P = \{P_1, P_2, \cdots, P_Z\}$ of size Z. For each dimension, it represents a topic and

denotes the user's influence over this topic. A query $q = \{q_1, q_2, \cdots, q_Z\}$ is also a Z dimension vector with $\sum q_i \leq 1$, where q_i denotes the importance of topic i in the promotion. Given query q, the propagation probability of the edge is computed as follows.

$$\mathbf{Pr}\,[e, q] = \sum_{i=1}^{Z} P_i \cdot q_i.$$

After calculating the probability of each edge, the topic-aware influence maximization problem transfers into general influence maximization problem. There are many papers [ABBB14, WC14, CFL$^+$15] focusing on processing the topic-aware influence maximization problem efficiently. If the query vectors' distributions are similar, the selected seed sets will be close to each other. Based on this motivation, in [ABBB14, WC14], the authors propose algorithms that utilize the information over some pre-sampled queries. [ABBB14] stores a set of results for certain sampled query vectors. Given a query, it first selects a set of results whose sampled query vectors are close to the input query. Then it merges these results and selects k nodes from it. Then closeness between two vectors is measure by the KL-divergence. The approach is very efficient, as it only needs to select from a very small set of nodes. However, it offers no guarantee for the quality of output results. [WC14] proposes two algorithms, namely Best Topic Selection (BTS) algorithm and Marginal influence Sort (MIS) algorithm, to generate the results from pre-computed seed sets. BTS and MIS can provide results with bounded approximation ratio based on two assumptions over the input data. The drawback of above two papers is that they either have no guarantee or have to make additional assumptions on the input data. To improve the efficiency without assumptions while providing guaranteed results, [CFL$^+$15] extends the MIA model and comes up with a best-first search algorithm. It can provide a result with $1 - 1/e$ approximation ratio under the MIA model.

In the above topic-aware influence maximization problem, the topic information

is conveyed through the probability of edges. While in some real applications, people tend to use keywords to represent their preferences, *i.e.,* topics. In [LZT15], the authors consider a target-based influence maximization problem, where each user is associated with a set of keywords, and the input query is also a set of keywords. The importance of each user is calculated as the TD-IDF score between the query keywords and the user's keywords. Then the problem is defined as a weighted influence maximization problem. The authors extend the RIS model to handle online queries efficiently. Through carefully analyzing the sample size required, the authors propose a disk-based index that can answer the online query with $1 - 1/e - \epsilon$ approximation ratio with high probability.

Location-Aware Influence Maximization. With the advancement of location-enabled device, each user is associated with a location in 2-dimensional space. Apart from the topic factor, users' location information become increasingly important for conducting a location-aware promotion. For example, to promote a local store via online social network, the selected seed sets are expected to be different when the promoted stores vary.

[LCF+14] is the first work considering users' location information in influence maximization problem. Specifically, users input a query region as the promoted area, and it tries to find a seed set that can maximize the influence only to the users in the query region. To meet the online requirement, the paper extends the MIA model, and develops three search algorithms by using different index structures. However, it is not an easy task to select a good query region to promote a local store. Since if the query region is too large, the selected seed set may influence a lot of users who are far away from the local store. Hence they may not come to the store due to the distance issue. While If the query region is too small, the number of users influenced might be very limited. Thus the result may not be satisfied.

It is more natural to consider the input as promoted location itself. In [ZPC+15], the authors consider that each user has a set of check-ins. Given a query location (*i.e.,* promoted store), it aims to infer the probability of each edge based on users' check-in history. For users who are more likely to attend the query location, the probability on the edge will be large. [WZZL16] considers the importance of the distance between each user and the query location. Users that are close to the query location will have high chance to attend. Based on this motivation, the paper proposes a distance-aware influence maximization model, which assigns each user a weight based on its distance to the query location. The authors also extend the MIA model for influence approximation and develop three pruning strategies to significantly reduce the searching space.

2.3 Graph Centrality

As a key concept in the social networks, centrality has been widely used to measure the importance of nodes in a social network. A social network is represented as a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges where $n = |V|$ and $m = |E|$. Many centrality metrics [WKF94, Fre78, Bra01] are proposed for different considerations to measure the importance of a node. For example, Google's PageRank is a variant of the eigenvector centrality. In this section, we review the techniques of computing nodes' closeness centralities and top-k closeness centrality computations.

2.3.1 Compute Closeness Centrality

The concept of closeness centrality is first formalized in [Bav48, Bav50] for the strongly connected graphs. As shown in Equation (2.3), it is defined as the inverse

of the average distance of a given node to all the other nodes in the graph.

$$c(v) = \frac{n-1}{\sum_{u \in V} d(v,u)}, \tag{2.3}$$

where $d(u,v)$ is the shortest path distance from v to u. There are also variant definitions of the closeness centrality, such as the definition in Equation (2.4), which considers an unconnected graph as the input.

$$c'(v) = \frac{(|V_c|-1)^2}{(n-1)\sum_{u \in V_v} d(v,u)}, \tag{2.4}$$

where V_v denotes the set of nodes that can be reached by v in G. In this thesis, we mainly focus on classic closeness centrality.

For different definitions of closeness centrality, we can compute the centrality of a node by running a single source shortest path query from the node. A major problem in calculating the closeness centrality is the scalability issue for large graphs, especially when we want to compute the closeness centrality for multiple nodes. Thus, many papers adopt sampling based methods to approximate the closeness centrality. For classic closeness centrality, the main problem is to compute the average of distance from the node to all the other nodes. Thus we can use the average distance to a set of sample nodes to estimate the closeness centrality [Ind99, Tho01, EW01], *i.e.,*

$$\hat{c}_s(v) = \frac{(n-1) \times l}{n \times \sum_{u \in \mathcal{L}} d(v,u)}, \tag{2.5}$$

where $\mathcal{L}$ denotes a set of sampled nodes and $l = |\mathcal{L}|$ is the sample size. It is easy to verify that $\frac{1}{c(v)} = \mathbb{E}[\frac{1}{\hat{c}_s(v)}]$. By applying Chernoff bound, we can derive the quality of the estimation as shown in Equation (2.6).

$$\Pr[|\frac{1}{c(v)} - \frac{1}{\hat{c}_s(v)}| \geq \epsilon\Delta] \leq \frac{2}{n^{2l\frac{\epsilon^2}{l \log n}(\frac{n-1}{n})^2}}, \tag{2.6}$$

where Δ is the diameter of the graph, *i.e.,* $\Delta = \max\{d(u,v)\}$.

In [Coh00], the authors adopt a pivot based method to approximate the shortest path distance. By extending the idea, we can derive a pivot based method to estimate the closeness centrality of a node. Suppose we randomly sample a set P of nodes, for each node $p \in P$, we compute its closeness centrality $c(p)$ by running a single source shortest path algorithm from it. Given a node v, to estimate its closeness centrality, we first find its nearest pivot p_i. Then we can use $\frac{1}{c(p_i)}$ to approximate $\frac{1}{c(v)}$. As we can see, $\frac{1}{c(v)} \in [\frac{1}{c(p_i)} - d(v, p_i), \frac{1}{c(p_i)} + d(v, p_i)]$. Nevertheless, in [CDPW14a] the authors show that with high probability the estimation $\frac{1}{c(p_i)} + d(v, p_i)$ is within a factor of 3 of the true value $\frac{1}{c(v)}$. In addition, when the number of pivots equals l, it is likely that the distance between p_i and v is one of the $\frac{\log n}{l}$ closest distances from v. Then the following equation holds with high probability.

$$\frac{1}{c(v)} \geq (1 - \frac{\log n}{l}) \cdot d(p_i, v).$$

However, the sampling based estimation is sensitive to the graph distribution, while the pivot based method may have a large error. To bridge the gap, in [CDPW14a, CCK15, Coh14], the authors propose a hybrid estimator by combing the two estimators together. Specifically, it aims to estimate the distance sum $S(v)$ from v to all the nodes, *i.e.,* $S(v) = \sum_{u \in V} d(v, u)$. It first samples a set of nodes, and computes the distance between the samples to the other nodes. To estimate the closeness centrality of v, it firstly identifies the closest sample p_v to v. Based on $d(v, p_v)$, it divides the nodes in $V \setminus \{v\}$ into three disjoint partitions, given an input parameter $\epsilon > 0$.

- $L(v, p_v)$ consists of the set of nodes whose distances are no greater than $\frac{d(v,p_v)}{\epsilon}$ from p_v.
- $HC(v, p_v)$ consists of the sampled nodes whose distances are greater than $\frac{d(v,p_v)}{\epsilon}$ from p_v.

- $H(v, p_v)$ consists of the set of nodes that does not belong to the samples but the distance are greater than $\frac{d(v,p_v)}{\epsilon}$ from p_v.

Based on the three partitions, the authors estimate the sum of distance from nodes in each partition to v. For nodes in $HC(v, p_i)$, the distance sum can be easily calculated, since their distance from v can be obtained exactly. For nodes in $L(v, p_i)$, it uses the samples in this partition to estimate the sum distance. For nodes in $H(v, p_i)$, it uses the pivot based method to estimate the sum of distance from v. Specifically, the estimation is formalized in Equation (2.7).

$$\hat{S}(v) = \frac{|L(v,p_v)|}{|L(v,p_v) \cap P|} \sum_{u \in L(v,p_v) \cap P} d(v,u) + \sum_{u \in HC(v,p_v)} d(v,u) + \sum_{u \in H(v,p_v)} d(p_v,u). \quad (2.7)$$

The authors show that the estimation has a bounded guarantee.

The error analyses for Equation (2.5) and (2.7) are for the estimation of a single node's closeness centrality. To bound the error for the estimation of all the nodes, we can apply the union bound and derive the sample size required.

2.3.2 Top-k Closeness Centrality

In this section, we review the techniques for finding top-k nodes with the largest closeness centrality score.

Exact Methods. To find the exact top-k result, naïvely we can run all pairs shortest path algorithm and compute the closeness centrality for each node. Finally k nodes with the largest closeness centralities are returned. The Floyd-Warshall algorithm [CSRL01] solves the all pairs shortest path problem based on dynamic programming, which requires $\Theta(|V|^3)$ time and $\Theta(|V|^2)$ space. The Floyd-Warshall algorithm is more suitable for dense graphs. For sparse graphs, [Joh77] proposes an approach by running Dijkstra's algorithm from each node. The time complexity is

$O(|V||E| + |V|^2 \log |V|)$ and the space complexity is $O(|V|^2)$. Based on the output of the algorithms, we can construct the shortest path for each pair of nodes. Since for top-k closeness centrality problem, only the distance between each pair of nodes is required, we can only store the pairwise distance to reduce the space cost.

The above two methods for all pairs shortest path computation are limited when the graph size is large. To accelerate the computation, we can leverage the shared information during computation. For example, suppose node v_1 has only one out going neighbour v_2. If we have computed the closeness centrality of v_2 or run the Dijkstra's algorithm from v_2, we can directly get the closeness centrality of v_1 based on the result of v_2, rather than computing the closeness centrality for v_1 from scratch. By adopting the idea of information sharing, [TKC+14] considers the multi-source breath first search (BFS) problem. Note that when the weight of each edge equals 1, we can compute the closeness centrality of a node by running a BFS from the node. In [OLH14], the authors provide a more efficient method that targets on the top-k closeness centrality problem. In addition to considering the shared information among computation, it can also efficiently derive the upper bound of a node's closeness centrality. Based on the bound, it can prune the nodes with small closeness centralities and reduce the computation cost. In [BBC+16], the authors consider an unweighed graph, and leverage the derived upper bound of a node's closeness centrality to significantly prune the node from exact computation.

Approximate Methods. A limitation of the exact methods of computing top-k closeness centrality is the scalability issue. To accelerate the search while still providing a bounded approximate result, some research are conducted to develop approximate methods in order to solve the top-k problem. The top-1 node is the node with the largest closeness centrality, which is also known as 1-median. In [Ind99, Tho01], the authors use the sampling based estimation method, *i.e.,*

Equation (2.5), to measure the closeness centrality of a node. In addition, it proposes a near linear time algorithm to find the top-1 node. Okamoto *et al.* [OCL08] further extend the sampling framework to efficiently find the top-k nodes. Specifically, it combines the approximation algorithm with the exact method to derive an algorithm of better time complexity.

Different from the problem introduced above, we consider a set of k nodes as a group instead of ranking them individually. We aim to find the set of k nodes which has the largest group closeness centrality as a whole. In [ZLTG14], the authors study the group closeness centrality maximization problem for an unweighed graph as well. For node v and a graph G with diameter of d, it utilizes the FM-sketch L_i to compute the number N_{L_i} of nodes that have distances from v no greater than i, $i \in [1, d]$. Then it approximates the distance sum of v to all the nodes as follows.

$$\hat{S}(v) = \sum_{i \in [1,d]} i \times (N_{L_i} - N_{L_{i-1}})$$

The accuracy of the estimation is decided by the size of FM-sketch. However, the techniques proposed are for the case when the weight of edge is 1, while our method is suitable for arbitrary weight setting.

Chapter 3

On Gapped Set Intersection Size Estimation

3.1 Overview

Set intersection size estimation is a fundamental operation in many areas, such as information retrieval, DBMS and data mining, etc. However, it is observed that only defining $\delta = 0$ in intersection size estimation is not enough, which we call the relation as equality set intersection. In some situations, it is more reasonable to define $b - a = \delta$ where $\delta > 0$, or $\delta \in [0, g)$.

In this chapter, we generalize the set intersection on integer sets to allow for "gaps" (i.e., $\delta > 0$). We define two primitives: the point gap constraint corresponds to a fixed gap of δ, and the range gap constraint corresponds to a gap of size no larger than δ. We are interested in methods to estimate these gapped set intersection size efficiently and accurately. This problem has many applications. For example, in sentiment analysis we extract different types of events, including a mention of a product, and an occurrence of sentiment (e.g., the word "`fantastic`").

We are usually interested in events that occur in a close vicinity. For instance, the *positions* of their occurrences in a document are within δ [HC05] or the timestamps of their occurrences in tweets are within δ [SEMP14]. We can model this as a gapped set intersection size estimation problem with a *range gap constraint* of δ.

Motivated by the examples, we define and study the problem of gapped set intersection size estimation (abbreviated as GSISE). For the estimation problem with point gap constraints, we propose a basic method that reduces the problem to the standard set intersection size estimation problem, which can be solved using the state-of-the-art sketch method. However, the index space is linear to the maximum query gap allowed. We improve it by judiciously selecting a subset of sketches to construct, and this reduces our sketch size from $O(N)$ to $O(\sqrt{N})$. The space cost of our approach is proved to be asymptotically optimal, while the time complexity remains the same. For the estimation problem with the range gap constraint, a baseline method is to reduce the problem into multiple estimation problems with different point gap constraints, and this requires estimation time linear in the gap size. We propose an extension of the bottom-k sketch to multiset and an unbiased estimator for inner product, we achieve an accurate and fast estimation method that is independent of the gap size. To demonstrate the use of our estimation methods, we consider the problem of finding highly correlated keywords from a large document collection. We design a new query processing method based on indexing the hash values in our sketches. Finally, we perform large-scale experimental evaluation using 500 million documents from the ClueWeb09 data collection and demonstrate the accuracy and efficiency of our proposed methods.

Contributions. The contributions of this chapter are summarized as follows.

- To the best of our knowledge, this is the first work to formally define the point and range gapped set intersection size estimation problems, which can

be used as primitive operations in a wide spectrum of applications.

- We design space and time efficient sketch and estimation methods for both types of estimation tasks. Our estimates are unbiased and have theoretical guarantees.

- We demonstrate the application of our technique for approximately mining top-K related keywords from large document collections. Our technique is especially useful in this application scenario where an exact solution requires orders of magnitudes larger space and time.

- Comprehensive experiments on the ClueWed09 dataset demonstrate the efficiency and accuracy of the proposed methods.

Organization of the chapter. The rest of the chapter is organized as follows. Section 3.2 defines the problem formally and introduces several useful techniques. Section 3.3 elaborates the estimation framework for point query and range query, together with their theoretical properties. Section 3.4 presents algorithm to top-K related keywords mining application. The experimental results are reported and analyzed in Section 3.5. We conclude the chapter in Section 3.6.

3.2 Background

In this section, we define two estimation problems, introduce existing work on estimating the size of set intersection using sketch, and finally list important notations used in the chapter.

3.2.1 Problem Definition

Given two sets S_A and S_B, their intersection is defined as $\{(a,b) \mid \texttt{eq(a,b)}, a \in S_A, b \in S_B\}$, where the predicate $\texttt{eq}(a,b)$ checks if a equals b. We can generalize the set intersection by considering other meaningful predicates. Specifically, with a *gap parameter* $\delta \geq 0$, we consider the following two predicates:

- $\texttt{PointGap}_\delta(a,b)$, which returns **true** if and only if $b - a = \delta$.
- $\texttt{RangeGap}_\delta(a,b)$, which returns **true** if and only if $0 \leq b - a \leq \delta$.

We call set intersection with these two new predicates collectively *gapped set intersection*. They are denoted as $S_A \cap^{=\delta} S_B$ (named *point gapped set intersection*) and $S_A \cap^{\leq\delta} S_B$ (named *range gapped set intersection*), respectively. Obviously, the standard set intersection is a special case of both types of gapped set intersection where $\delta = 0$.

In this chapter, we study space and time-efficient methods to estimate the results size of these two types of gapped set intersection. Motivated by the applications we aim at, we consider sets whose elements are integers.

Definition 3.1 (GSISE). *Given two sets S_A and S_B, and a gap parameter $\delta \in [0, N]$ where N is a predefined maximum gap value. The* Point *and* Range *Gapped Set Intersection Size Estimation Problem is to estimate $|S_A \cap^{=\delta} S_B|$ and $|S_A \cap^{\leq\delta} S_B|$, respectively. They are abbreviated as* point estimation *and* range estimation *hereafter.*

Example 3.1. *Let $S_A = \{1,3,4,7\}$ and $S_B = \{2,5,6,8\}$, and $\delta = 2$. Under the point gap constraint, $S_A \cap^{=2} S_B = \{(3,5),(4,6)\}$. Hence its size is* 2. *Under the range gap constraint, $S_A \cap^{\leq 2} S_B = \{(1,2),(3,5),(4,5),(4,6),(7,8)\}$. Hence its size is* 5.

Discussions. In the above definitions, we only need to consider $\delta \geq 0$. This is because $S_A \cap^{=\delta} S_B = S_B \cap^{=-\delta} S_A$ (this also holds for range gapped set intersection too).

Many other types of interesting gapped set intersections can be defined using our point and range gapped set intersection as primitives. For example, consider the predicate $\texttt{RangeWithin}_{\delta_1,\delta_2}(\texttt{a}, \texttt{b})$ with two range parameters $0 \leq \delta_1 < \delta_2$, which checks if $\delta_1 \leq a - b \leq \delta_2$. It is easy to see that its query result is exactly the difference of two range gapped set intersection, i.e., $(S_A \cap^{\leq \delta_2} S_B) \setminus (S_A \cap^{\leq \delta_1} S_B)$.

In a similar fashion, we can derive point gapped set intersection based on range gapped set intersection, and vice versa. Nevertheless, we still consider them as two separate primitives, as each of them can model different applications respectively, and we will propose related but different estimation methods and optimizations for them.

3.2.2 Bottom-k Sketch

The state-of-the-art method to estimate the size of set intersections is the bottom-k sketch [CK07, CK08, CK09]. It is a lightweight sketch that supports efficient update.

We use $sk(S_A)$ to denote the bottom-k sketch for a set S_A, which is a set of k hash values. To construct the sketch, we apply a random hash function h to every element $a \in S_A$, and keep the k minimum hash values. We also assume the codomain of h is sufficiently large such that we can safely assume that there is no collision.

An important property of the bottom-k sketch is that it is closed under set union operation. Specifically, we can directly compute the bottom-k sketch of the union of two sets from their respective bottom-k sketches. The resulting sketch is

called *short combination sketch* [CK09]: $scs(sk(S_A), sk(S_B)) = \{v \mid v \in sk(S_A) \cup sk(S_B), v < \min(sk(S_A)^{(k)}, sk(S_B)^{(k)})\}$, where the notation $S^{(k)}$ denotes the k-th smallest value in the set S.

To estimate the intersection size of two sets S_A and S_B from their respective sketches, we compute

$$t = \frac{|sk(S_A) \cap sk(S_B) \cap scs(sk(S_A), sk(S_B))|}{\min\{sk(S_A)^{(k)}, sk(S_B)^{(k)}\}}. \tag{3.1}$$

[CK09] proved that t is an unbiased estimator of $|S_A \cap S_B|$ that can be computed efficiently. Besides, it is shown to have smaller variance compared with the Minhash [BCFM98] method.

3.2.3 Notations

Table 3.1 lists notations frequently used in the chapter.

Table 3.1: Summary of Notations

Symbol	**Explanation**
S_{id}	a set of integers
M_{id}	a multiset; each element has its multiplicity
S_{id}^{+d}	a shifted set by distance d
$\{d_1, d_2, \ldots\}$	the index of a multiset generated by merging the set shifted by d_1, d_2, ...
$S^{(i)}$ or $M^{(i)}$	i-th smallest value in the set S or the multiset M
N	the maximum gap
δ	the gap parameter; $\delta \in [0, N]$
k	number of hash values in a bottom-k sketch
$sk(S)$	bottom-k sketch of the set S
$msk(M)$	multiset bottom-k sketch of the multiset M
rus	range union sketch of $msk(M_A)$ and $msk(M_B)$

3.3 Estimation Methods for Gapped Set Intersection Size

In this section, we introduce solutions to point and range estimation problems. The naïve index structure and our intuition are explained in Section 3.3.1, followed by technique details of GSISE methods in Section 3.3.2 and 3.3.3.

3.3.1 A Baseline Method for Point Estimation

Point estimation is a challenging problem due to the following reasons:

- Almost all the set intersection size estimation methods are based on random hash functions. Therefore, given a gap value δ, and two *different* hash values $h(x+\delta)$ and $h(y)$, it is almost impossible to infer if $x+\delta$ is equal to y.

- While locality sensitive hash functions [I+98] do preserve locality probabilistically, the intersection size is highly sensitive to the point gap threshold. It is often the case that $|S_A \cap^{=\delta} S_B|$ and $|S_A \cap^{=\delta+1} S_B|$ differ substantially. For instance, in Example 3.1, $|S_A \cap^{=0} S_B|$ is 0 while $|S_A \cap^{=1} S_B|$ is 3. Therefore, even a small approximation in the gap may result in a large estimation error.

Therefore, we first propose a baseline method, which reduces the point estimation problem to a standard set intersection size estimation problem, which in turn can be solved using the state-of-the-art method, such as the method based on bottom-k sketches.

Our reduction is based on the notion of *shifted sets*. Given a set $S_A = \{a_1, a_2, \ldots, a_n\}$ and integer parameter d, we define the shifted set with shift d as follows: $S_A^{+d} = \{a_i + d \mid a_i \in S_A\}$.

Given the maximum gap size N, we compute $N+1$ bottom-k sketches by shifting S_A: the i-th bottom-k sketch is built for the shifted set S_A^{+i} $(0 \leq i \leq N)$. To perform the estimation for $S_A \cap^{=\delta} S_B$, we retrieve the sketches of $S_A^{+\delta}$ and S_B^{+0}, and perform the estimation using the bottom-k estimation procedure (i.e., Eq. (3.1)).

Table 3.2: The Random Hash Function h

x	0	1	2	3	4	5	6	7	8
$h(x)$	0.47	0.26	0.32	0.84	0.74	0.79	0.22	0.42	0.95
x	9	10	11	12	13	14	15	16	17
$h(x)$	0.48	0.68	0.89	0.16	0.63	0.37	0.53	0.15	0.21

Example 3.2. *Table 3.3 shows the sketches built by the baseline method for two sets S_A and S_B, where the bottom-k sketch size is 3 and maximum gap N is 9.*

To perform the point estimate for $\delta = 5$, we load $sk(S_A^{+5})$ and $sk(S_B^{+0})$, the estimate according to Eq. (3.1) is $1/\min(0.48, 0.79) = 2.08$. The actual point gapped intersection size is 2.

Table 3.3: $N+1$ bottom-k Sketches Built for $S_A = \{1, 3, 4, 7\}$ and $S_B = \{2, 5, 6, 8\}$ using the Hash Function in Table 3.2. $N = 9$.

shift	**Sketches for S_A**	**Sketches for S_B**
0	$sk(S_A^{+0}) = \{0.26, 0.42, 0.74\}$	$sk(S_B^{+0}) = \{0.22, 0.32, 0.79\}$
⋮	⋮	⋮
5	$sk(S_A^{+5}) = \{0.16, 0.22, 0.48\}$	$sk(S_B^{+5}) = \{0.42, 0.63, 0.68\}$
⋮	⋮	⋮
9	$sk(S_A^{+9}) = \{0.15, 0.16, 0.63\}$	$sk(S_A^{+9}) = \{0.21, 0.37, 0.53\}$

Obviously, this baseline method achieves the same accuracy guarantee as the standard bottom-k sketch [BHR+07, CK07, PSW14], and the estimation time is $O(k)$. The main problem is its space complexity of $O(N \cdot k)$ for each set, which is not optimal.

3.3.2 Improved Point Query Estimation

In this subsection, we seek to improve the space complexity per set from $N \cdot k$ to $\sqrt{N} \cdot k$, while still maintaining the same $O(k)$ estimation time. We also show that this space complexity is asymptotically optimal with an approximation ratio of $\sqrt{2}$.

We observe that we can generate sketches for a *judiciously chosen subset* of all possible shifted sets, to ensure that can still find two appropriate sketches for two sets to perform the point estimation for any $\delta \in [0, N]$.

Let $\lambda := \lceil \sqrt{N} \rceil$. For a set S_A, we generate 2λ sketches for S_A shifted by $i \in I$, where

$$I = \{0, 1, 2, \cdots, \lambda - 1\} \cup \{i \cdot \lambda \mid i \in [1, \lambda]\} \tag{3.2}$$

Algorithm 1 describes the estimation procedure. Lines 2–5 compute the correct offsets (also called indices) of the sketches of S_A and S_B. Then Line 6 performs the estimation with the corresponding bottom-k sketches.

Algorithm 1: PointQueryOnlineEstimation

Input : δ: query gap. N: maximum gap.

Output: An estimate of $|S_A \cap^{=\delta} S_B|$

1 $\lambda \leftarrow \lceil \sqrt{N} \rceil$;

2 **if** $\delta \pmod{\lambda} = 0$ **and** $\delta \neq N$ **then**

3 $\quad i_A \leftarrow \lambda + \delta; \quad i_B \leftarrow \lambda;$

4 **else**

5 $\quad i_A \leftarrow \lceil \delta/\lambda \rceil \cdot \lambda; \quad i_B \leftarrow \lceil \delta/\lambda \rceil \cdot \lambda - \delta;$

6 **return** Bottom-k-Estimate $(sk(S_A^{+i_A}), sk(S_B^{+i_B}))$

Example 3.3. *Continue the previous example.* $\lambda = \lceil \sqrt{N} \rceil = 3$. *In our improved*

method, the following sketches are generated for every set S

$$sk(S_i^{+0}), sk(S_i^{+1}), sk(S_i^{+2}), sk(S_i^{+3}), sk(S_i^{+6}), sk(S_i^{+9}).$$

For point estimation with $\delta = 5$*, we compute* $i_A = 6$ *and* $i_B = 1$ *according to Algorithm 1. Then we load* $sk(S_A^{+6})$ *and* $sk(S_B^{+1})$ *and perform the estimation.*

The correctness of Algorithm 1 depends on two facts:

- Gapped set intersection size is shift-invariant as shown in Lemma 3.1, and
- $\forall \delta \in [0, N]$, we can always locate the i_A and i_B from the index set (Equation (3.2)) such that $i_A - i_B = \delta$.

Lemma 3.1. $\forall d$, $|S_A \cap^{=\delta} S_B| = |S_A^{+d} \cap^{=\delta} S_B^{+d}|$.

The improved method has a $O(\sqrt{N} \cdot k)$ space complexity per set, $O(k)$ estimation time, and the same estimation quality guarantees as the bottom-k sketch.

Asymptotic Space Optimality and Approximation Ratio

We show that our improved method is asymptotically space optimal in this reduction framework. To facilitate the analysis, a computational model is formalized below in Definition 3.2.

Definition 3.2. *Given set* $G = \{0, 1, 2, \cdots, N\}$*, where* N *is the maximum gap predefined. We want to find an integer set* P *with minimum cardinality, such that* $\forall g \in G$*, there exist* $i, j \in P$ *satisfying* $i - j = g$.

Property 3.1. *The lower bound of* $|P|$ *is* $\Omega(\sqrt{N})$.

Proof. First there are no duplicate elements in P, otherwise $|P|$ cannot reach the minimum since it is allowed to choose two identical elements from P to get $0 \in G$.

Now that all elements in P are different, we can construct a mapping f from $i-j$ to $G \setminus \{0\}$. It is easy to see the lower bound will be achieved when the mapping is bijective. Hence, the number of all possible choices of $i-j$ is $\binom{|P|}{2}$. Since $\binom{|P|}{2} \geq N$, we have $|P| = \Omega(\sqrt{N})$. □

Corollary 3.1. *The number of sketches constructed in our improved method is within a factor of* $\sqrt{2}$ *of the optimal solution.*

3.3.3 Range Estimation

Basic Method for Range Estimation

Our basic method for range estimation is to reduce a range estimation to multiple point estimations, due to the following equivalence.

Theorem 3.1. $S_A \cap^{\leq \delta} S_B = \bigcup_{i=0}^{\delta} (S_A \cap^{=i} S_B)$. *In addition,* $\forall i \neq j, (\{S_A \cap^{=i} S_B\}) \cap (\{S_A \cap^{=j} S_B\}) = \emptyset$.

Theorem 3.1 reveals that the range gapped set intersection results can be partitioned into disjoint subsets, each is the result of a point gapped set intersection. Taking the cardinality on both sides of the equation and we have the following Corollary.

Corollary 3.2. $|S_A \cap^{\leq \delta} S_B| = \sum_{i=0}^{\delta} |S_A \cap^{=i} S_B|$.

Corollary 3.2 enables us to sum up $\delta + 1$ point estimation results to answer a range estimation. While this does not increase the space complexity, the estimation time is linear in the range δ. When δ is large, the estimation time grows quickly, which is not desirable.

Merge of Shifted Sets

We introduce several essential concepts, which enable us to present an observation using an example. This relates the range estimation to the problem of estimating the inner product of multisets, each obtained by merging shifted sets in a particular pattern.

We define a multiset M as a set of elements, each associated with its multiplicity, i.e., $M = \{a_1 : m_1, a_2 : m_2, \ldots, a_n : m_n\}$. $\mathsf{elems}(M)$ returns all the elements in the multiset M, i.e., $\{a_1, a_2, \ldots, a_n\}$. The multiplicity of an element e in M is denoted as $\mathsf{cnt}_M(e)$. Note that a set is just a special case of a multiset where all the multiplicities equal 1.

Let U be the universe of all elements. Each multiset can be implicitly cast into a $|U|$-dimensional vector where the dimension values are the multiplicities of the corresponding elements (default to 0). Hence, we can define the *inner product* of two multisets as

$$\langle M_A, M_B \rangle = \sum_{e \in \mathsf{elems}(M_A) \cap \mathsf{elems}(M_B)} \mathsf{cnt}_{M_A}(e) \cdot \mathsf{cnt}_{M_B}(e).$$

Now, we can illustrate an important observation that leads to our improved range estimation in Example 3.4.

Example 3.4. *Consider the same instance in Example 3.3 and we want to estimate* $S_A \cap^{\leq \delta} S_B$. *Table 3.4(a) enumerates for all possible* $\delta \in [0, 8]$ *the shifted* S_A *and* S_B *that will be used for set intersection with point gap* δ. *For example, the cell with green background is for* $\delta = 5$*; it is associated with* S_A^{+6} *and* S_B^{+1}. *This means* $|S_A \cap^{=5} S_B| = |S_A^{+6} \cap S_B^{+1}|$.

Now consider the range gapped set intersection with $\delta = 5$. *Corollary 3.2 shows that the result size* $|S_A \cap^{\leq 5} S_B|$ *equals the sum of the size of 6 point gapped set intersections, with the gap constraint between 0 and 5. By looking at Table 3.4(a),*

Table 3.4: Illustration of Shift Sets Used for Point/Range Estimation for $\delta \in [0, 8]$

(a) Shifted Sets (Illustrating $\delta = 5$)

S_A^{+9}	8	7	6
S_A^{+6}	5	4	3
S_A^{+3}	2	1	0
	S_B^{+1}	S_B^{+2}	S_B^{+3}

(b) Merged Shifted Sets (Illustrating $\delta = 7$)

$\biguplus_{i\in\{3,6,9\}} S_A^{+i}$	8	7	6	S_A^{+9}
$\biguplus_{i\in\{3,6\}} S_A^{+i}$	5	4	3	S_A^{+6}
$\biguplus_{i\in\{3\}} S_A^{+i}$	2	1	0	S_A^{+3}
	$\biguplus_{i\in\{1,2,3\}} S_B^{+i}$	$\biguplus_{i\in\{2,3\}} S_B^{+i}$	$\biguplus_{i\in\{3\}} S_B^{+i}$	

we can see the latter is equivalent to $\langle M_A, M_B \rangle$*, where*

$$M_A = (S_A^{+3} \uplus S_A^{+6}), \qquad M_B = (S_B^{+1} \uplus S_B^{+2} \uplus S_B^{+3})$$

Note that we need to use multiset union (also called ***merge*** *in this chapter, denoted as* $\uplus$*) and the inner product, as there may be potential duplicate elements. Obviously, if we can estimate the inner product of multisets, then we can perform one estimation rather than six point estimations.*

By considering all possible δ values in Example 3.4, we can observe the pattern where multiple shifted sets are merged. To simplify the notation, we use the set of shift values as an identifier (or index) for the merged multiset, i.e., if $M = S_A^{+i_1} \uplus S_A^{+i_2} \ldots \uplus S_A^{+i_j}$, then we say M's **index** is $\{i_1, i_2, \ldots, i_j\}$ and it uniquely identifies M.

Let $\lambda = \lceil \sqrt{N} \rceil$, and $i \geq 1$.

- For shift values within $[1, \lambda]$, we need to merge its suffixes, i.e., generating indices $\{i, i+1, \ldots, \lambda\}$, for $1 \leq i < \lambda$.

- For shift values within $[\lambda, \lambda^2]$, we need to merge its prefixes, i.e., generating indices $\{\lambda, 2\lambda, \ldots, i \cdot \lambda\}$, for $1 < i \leq \lambda$.

Estimating the Inner Product of Two Multisets

As motivated in Example 3.4, we need to estimate the inner product of two multisets. While there are many alternative methods (such as Tug-of-War [AMS96] and Count-Min [CM04] sketches), we observe that the input multisets are always those shifted sets generated for the point estimation task, which means each of them has already its bottom-k sketch built or maintained. Therefore, we develop an estimator by *extending the bottom-k sketch* as follows.

Firstly, we define our **multiset bottom-k sketch** for a multiset obtained by merging multiple shifted sets, each with its own bottom-k sketch. Let $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ denote a set of shifted sets, each with its bottom-k sketch $sk(S_i)$. Let $M = \uplus_{i=1}^{n} S_i$. M's multiset bottom-k sketch, denoted by $msk(M)$, is obtained by a *truncated merge* of the sketches, i.e.,

1. first merging $sk(S_i)$ into a multiset M', and
2. then keeping only the k smallest elements and their multiplicities in M'.

Figure 3.1 illustrates the relationship between a multiset bottom-k sketch and its constituent bottom-k sketches.

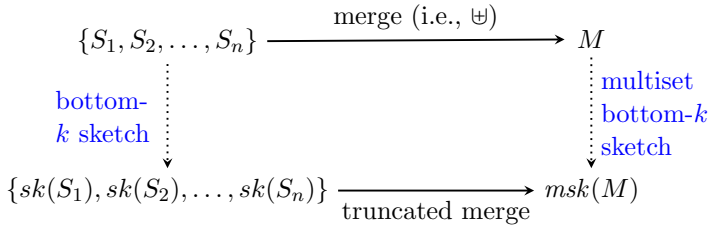


Figure 3.1: From bottom-k Sketches to a Multiset bottom-k Sketch

Given two multiset bottom-k sketches $msk(M_A)$ and $msk(M_B)$, their *range union sketch*, denoted by $rus(msk(M_A), msk(M_B))$, is a multiset that contains all the hash values in $\mathsf{elems}(msk(M_A)) \uplus \mathsf{elems}(msk(M_B))$ that are smaller than $\min(msk(M_A)^{(k)}, msk(M_B)^{(k)})$, as well as the sum of their multiplicities in $msk(M_A)$ and $msk(M_B)$. Given $msk(M_A)$ and $msk(M_B)$, we propose an unbiased estimator of $\langle M_A, M_B \rangle$, as shown in Theorem 3.2. Furthermore, it can be shown that, by using the range union sketch, our method takes advantage of all the information available in $msk(M_A)$ and $msk(M_B)$ to arrive at the best possible estimation.

Theorem 3.2. *Given two multiset bottom-k sketches $msk(M_A)$ and $msk(M_B)$, let $rus_\cap = \mathsf{elems}(msk(M_A)) \cap \mathsf{elems}(msk(M_B)) \cap \mathsf{elems}(rus(msk(M_A), msk(M_B)))$, then*

$$\hat{t}_r = \frac{\sum_{e \in rus_\cap} \left(\mathsf{cnt}_{msk(M_A)}(e) \cdot \mathsf{cnt}_{msk(M_B)}(e) \right)}{\min\{msk(M_A)^{(k)}, msk(M_B)^{(k)}\}}. \tag{3.3}$$

is an unbiased estimator of $\langle M_A, M_B \rangle$.

Proof. Define *adjusted multiplicity* for each $e \in \mathsf{elems}(M_A) \cap \mathsf{elems}(M_B)$ to be:

$$a_e = \begin{cases} \frac{\mathsf{cnt}_{msk(M_A)}(e) \cdot \mathsf{cnt}_{msk(M_B)}(e)}{rus^{(l)}} & \text{, if } e \text{ is sampled in } rus_\cap \\ 0 & \text{, otherwise.} \end{cases} \tag{3.4}$$

where $rus^{(l)} = \min\{msk(M_A)^{(k)}, msk(M_B)^{(k)}\}$, i.e., it is the l-th smallest hash values in rus. Then we can write $\hat{t}_r$ as $\sum_{e \in rus_\cap} \frac{\mathsf{cnt}_{msk(M_A)}(e) \cdot \mathsf{cnt}_{msk(M_B)}(e)}{rus^{(l)}}$. The expec-

tation is

$$\begin{aligned}\mathbf{E}\left[\hat{t}_r\right] &= \mathbf{E}\left[\sum_{e\in rus_\cap} \frac{\mathsf{cnt}_{msk(M_A)}(e)\cdot \mathsf{cnt}_{msk(M_B)}(e)}{rus^{(l)}}\right] \\ &= \sum_{e\in \mathsf{elems}(M_A)\cap\mathsf{elems}(M_B)} \mathbf{E}\left[a_e\right] \\ &= \sum_{e\in \mathsf{elems}(M_A)\cap\mathsf{elems}(M_B)} \mathsf{cnt}_{msk(M_A)}(e)\cdot \mathsf{cnt}_{msk(M_B)}(e) \\ &= \sum_{e\in \mathsf{elems}(M_A)\cap\mathsf{elems}(M_B)} \mathsf{cnt}_{M_A}(e)\cdot \mathsf{cnt}_{M_B}(e).\end{aligned}$$

The last step is because rus keeps *complete* multiplicity information for each underlying set element included in rus. *Complete* means if element e is sampled in $rus_\cap$, it holds that $\mathsf{cnt}_{msk(M)}(e) = \mathsf{cnt}_M(e)$. Since we are doing consistent uniform sampling in $\mathsf{elems}(M_A) \cup \mathsf{elems}(M_B)$. Therefore, $\mathbf{E}\left[\hat{t}_r\right] = \langle M_A, M_B\rangle$. □

Furthermore, by setting k to an appropriate value, we can achieve a probabilistic guarantee for $\hat{t}_r$, as shown in Theorem 3.3.

Theorem 3.3. *Let* $\mu = \langle M_A, M_B\rangle$. *For any given* ϵ *and* ρ, *by setting* $k = \min\{\max\{k_1, k_2\},\ \max\{|\mathsf{elems}(M_A)|,\ |\mathsf{elems}(M_B)|\}\}$, *where* k_1 *satisfies Eq.* (3.5) *and* $k_2 = \frac{(|\mathsf{elems}(M_A)|+2)N}{\mu}\cdot\frac{2+\epsilon}{\epsilon^2}\ln\frac{4}{\rho}$, *we can guarantee that* $\mathbf{Pr}\left[|\hat{t}_r - \mu| \le \epsilon\mu\right] \ge 1-\rho$.

Proof. We first define two propositions (a) and (b). Let $X_i = a_i \cdot \frac{k}{(|\mathsf{elems}(M_A)|+2)N}$ and e_i be the corresponding element, a_i as defined in Eq. (3.4), and k_1 and k_2 as defined in the Theorem.

- Proposition (a): When $k \ge k_1$, $\mathbf{Pr}\left[X_i > 1\right] \le \frac{\rho}{2}$.
- Proposition (b): When $k \ge k_2$, $\mathbf{Pr}\left[|\hat{t}_r - \mu| \ge \epsilon\mu\right] \le \frac{\rho}{2}$.

If both of them are proved, then when $k \ge \max\{k_1, k_2\}$, we know that $\mathbf{Pr}\left[|\hat{t}_r - \mu| \ge \epsilon\mu\right] \le \rho$ based on the Union Bound.

Proof of Proposition (a). Without loss of generality, we assume $|\mathsf{elems}(M_A)| > |\mathsf{elems}(M_B)|$. Given ρ, let k_1 be solution to Eq. (3.5):

$$\int_0^{\frac{k}{|\mathsf{elems}(M_A)|+2}} \frac{t^{k-1}(1-t)^{|\mathsf{elems}(M_A)\cup\mathsf{elems}(M_B)|-k}}{B(k, |\mathsf{elems}(M_A)\cup\mathsf{elems}(M_B)|-k+1)}dt \leq \frac{\rho}{2}, \tag{3.5}$$

where $B(k, |\mathsf{elems}(M_A)\cup\mathsf{elems}(M_B)|-k+1)$ is the Beta function. It also holds:

$$\begin{aligned}
&\mathbf{Pr}\left[X_i > 1\right] \\
&= \mathbf{Pr}\left[\frac{\mathsf{cnt}_{msk(M_A)}(e_i)\cdot\mathsf{cnt}_{msk(M_B)}(e_i)}{msk(M_A)^{(k)}} > \frac{(|\mathsf{elems}(M_A)|+2)N}{k}\right] \\
&\leq \mathbf{Pr}\left[msk(M_A) < \frac{k}{|\mathsf{elems}(M_A)|+2}\right].
\end{aligned} \tag{3.6}$$

The last step is because $\mathsf{cnt}_{msk(M_A)}(e_i)\cdot\mathsf{cnt}_{msk(M_B)}(e_i) \leq N$. Besides, when $k > k_1$, Eq. (3.5) is equivalent to:

$$\mathbf{Pr}\left[msk(M_A)^{(k)} < \frac{k}{|\mathsf{elems}(M_A)|+2}\right] < \frac{\rho}{2}. \tag{3.7}$$

Since we are doing uniform random sample in the space of $\mathsf{elems}(M_A)\cup\mathsf{elems}(M_B)$, where $msk(M_A)^{(k)}$ is the k-th order statistics therein.

Thus from Eq. (3.6) and (3.7), we know when $k > k_1$, $\mathbf{Pr}\left[X_i > 1\right] \leq \frac{\rho}{2}$.

Proof of Proposition (b). Now that we can guarantee $X_i \in [0,1]$ almost surely by proving Proposition (a), we can define $X = \sum X_i$ and apply the Chernoff bound to X.[1]

Let $k_2 = \frac{(|\mathsf{elems}(M_A)|+2)N}{\mu}\cdot\frac{2+\epsilon}{\epsilon^2}\ln\frac{4}{\rho}$. When $k > k_2$, we have

$$2\exp\left(-\frac{\epsilon^2}{2+\epsilon}\frac{\mu k}{(|\mathsf{elems}(M_A)|+2)N}\right) \leq \frac{\rho}{2}.$$

[1]Since bottom-k belongs to sampling without replacement strategy, it will cause negative associations between each sample value [DR96]. Nevertheless, according to [DR96], Chernoff bounds are still applicable to sums of random variables with negative associations.

According to the Chernoff bound, we have

$$\begin{aligned}&\mathbf{Pr}\left[|\hat{t}_r - \mu| \geq \epsilon\mu\right]\\ &\leq 2\exp\left(-\frac{\epsilon^2}{2+\epsilon}\cdot\frac{\mu k}{(|\mathsf{elems}(M_A)|+2)N}\right) \leq \frac{\rho}{2}.\end{aligned}$$

Then from Union Bound, it guarantees that when $k \geq \max(k_1, k_2)$, $\mathbf{Pr}\left[|\hat{t}_r - \mu| \geq \epsilon\mu\right] \leq \delta$. □

Improved Range Estimator

In our improved methods, given a maximum gap N, let $\lambda = \lceil\sqrt{N}\rceil$. We generate three categories of sketches:

- Category I: $\lambda + 1$ *bottom-k sketches* for shifts $i \in \{j\lambda \mid 0 \leq j \leq \lambda\}$
- Category II: $\lambda - 1$ *multiset bottom-k sketches* with indices $\{\lambda, 2\lambda\}$, $\{\lambda, 2\lambda, 3\lambda\}$, $\cdots$, $\{\lambda, 2\lambda, \ldots, \lambda^2\}$.
- Category III: $\lambda - 1$ *multiset bottom-k sketches* with indices $\{\lambda, \lambda-1\}$, $\{\lambda, \lambda-1, \lambda-2\}$, $\cdots$, $\{\lambda, \lambda-1, \ldots, 1\}$.

Table 3.4(b) illustrates Category I sketches on the right-hand side, Category II sketches on the left-hand side, and Category III sketches on the bottom side, for the running example.

For any range $[0, \delta]$, it can always be decomposed to at most two sub-ranges. For example, Table 3.4(b) illustrates that $[0, 7]$ is decomposed into

- $[0, 5]$. This corresponds to $S_A \cap^{\leq 5} S_B$, and can be answered by estimating the size of $\langle \biguplus_{i\in\{3,6\}} S_A^{+i}, \biguplus_{i\in\{1,2,3\}} S_B^{+i}\rangle$.
- $[6, 7]$. This corresponds to $S_A \cap^{[6,7]} S_B$, and can be answered by estimating the size of $\langle S_A^{+9}, \biguplus_{i\in\{2,3\}} S_B^{+i}\rangle$.[2]

Algorithm 2: DecomposeRange

Input : δ: query gap. N: maximum gap.

Output: An array of indices pairs.

$\lambda \leftarrow \lceil \sqrt{N} \rceil$;
if $\delta < \lambda$ **or** $(\delta + 1) \mod x = 0$ **then**
  $i_A \leftarrow \{i \cdot \lambda \mid i \in [1, \max(\lceil \delta/\lambda \rceil, 1)]\}$;
  $i_B \leftarrow \{\lambda - i \mid i \in [0, \delta - \lfloor \delta/\lambda \rfloor \cdot \lambda]\}$;
  return $\{(i_A, i_B)\}$
else
  $i_A^l \leftarrow \{i \cdot \lambda \mid i \in [1, \lfloor (\delta + 1)/\lambda \rfloor]\}$;
  $i_B^l \leftarrow \{i \mid i \in [1, \lambda]\}$;
  if $\delta = N$ **then**
    $i_A^r \leftarrow N$;
    $i_B^r \leftarrow 0$;
  else
    $i_A^r \leftarrow \lambda \cdot \lceil (\delta + 1)/\lambda \rceil$;
    $i_B^r \leftarrow \{\lambda - i \mid i \in [0, \delta \mod \lambda]\}$;
  return $\{(i_A^l, i_B^l), (i_A^r, i_B^r)\}$

Algorithm 3: RangeQueryOnlineEstimation

Input : δ: query gap. N: maximum gap.

Output: $\hat{t}_r$: an estimate for $S_A \cap^{\leq\delta} S_B$.

1 $arr \leftarrow$ DecomposeRange(δ) ;

2 $\hat{t}_r \leftarrow 0$;

3 **for each** $(i_A, i_B) \in arr$ **do**

4 Retrieve S_A's sketch based on the index of i_A into $msk(M_A)$ and retrieve S_B's sketch for i_B into $msk(M_B)$;

5 $\hat{t}_r \leftarrow \hat{t}_r +$ Estimate($msk(M_A), msk(M_B)$); `/* Perform estimation using two multiset bottom-`k` sketches based on Theorem 3.2 */`;

6 **return** $\hat{t}_r$

We use Algorithm 3 to perform a range estimation with the gap constraint δ. In Line 1, it calls the DecomposeRange (δ) to decompose the range $[0, \delta]$ into one or two parts, utilizing the multiset bottom-k sketches. The returned results are one or two pairs of indices to these sketches. Lines 3–5 retrieve the corresponding sketches and perform the estimation based on Eq. (3.3) in Theorem 3.2.

The space complexity of the sketch for the range estimation is $O(\sqrt{N} \cdot k)$. And the time complexity for the estimate is $O(k)$. Thanks to Theorem 3.3, by setting the proper k, our estimation has at most ϵ relative error with probability at least $1 - \rho$.

Example 3.5. *We run Algorithm 3 for the same running example given before. Given* $\delta = 7$*, we want to estimate* $|S_A \cap^{\leq 7} S_B|$*. Line 1 decomposes* $[0, 7]$ *into two parts of indices, which is* $arr = \{(\{3, 6\}, \{1, 2, 3\}), (\{9\}, \{2, 3\})\}$*. For* $(\{3, 6\}, \{1, 2, 3\})$*, Line 4 retrieves* $msk(S_A^{+3,6}) = \{0.22 : 1, 0.42 : 2, 0.48 : 1\}$ *and*

[2]We deem a bottom-k sketch as a special multiset bottom-k sketch where the multiplicities are all set to 1.

Table 3.5: Multiset bottom-3 Sketch

indices	$msk(M_A)$	$msk(M_B)$
$\{9\}$	$\{0.15:1, 0.16:1, 0.63:1\}$	$\{0.21:1, 0.37:1, 0.53:1\}$
$\{2,3\}$	$\{0.22:2, 0.42:1, 0.48:1\}$	$\{0.42:1, 0.48:1, 0.68:1\}$
$\vdots$	$\vdots$	$\vdots$
$\{3,6\}$	$\{0.22:1, 0.42:2, 0.48:1\}$	$\{0.16:1, 0.37:1, 0.48:1\}$
$\{1,2,3\}$	$\{0.22:2, 0.32:1, 0.42:1\}$	$\{0.22:1, 0.42:2, 0.48:2\}$

$msk(S_B^{+1,2,3}) = \{0.22:1, 0.42:2, 0.48:2\}$ *from Table 3.5. Then Line 5 calculates* $\hat{t}_r$ *according to Theorem 3.2. It returns* $\hat{t}_r$ *to be* $^{(1\cdot 1+2\cdot 2)}/_{0.48} = 10.42$. *Similar steps go for the second part indexed by* $(\{9\},\{2,3\})$, *while there is no matching for this part. Finally, the algorithm returns* $\hat{t}_r$ *to be* 10.42. *The actual* $|S_A \cap^{\leq 7} S_B|$ *is* 11.

3.4 Applications

The proposed methods have many important applications, e.g. top-K related keywords mining, query optimization for search engine, and system troubleshooting in log analysis. In this section, we present the details of efficient mining top-K related keywords from a document collection.

Let $\mathcal{V}$ be the vocabulary of the document collection. For ease of illustration, we concatenate all documents into one single document D with suitable padding of out-of-vocabulary keywords. For each keyword $v \in \mathcal{V}$, we create a set S_v, which consists of all the positions in D where it occurs. Let query S_Q be a set of positions. For any given keyword v, we can measure its correlation with the query by counting the number of occurrences of v in a δ-vicinity of any position in S_Q. The top-K related keywords problem is to find the K keywords in $\mathcal{V}$ that has the highest correlation.

Solving the problem exactly requires either intersecting all keywords in $\mathcal{V}$ or retrieving the δ-vicinity centered at positions in S_Q. Neither method scales well with large document collections.

To solve the problem approximately, we can apply range estimation to estimate the correlation, then return the K keywords with largest estimated size. Nevertheless, this method is still time consuming, as it is linear to vocabulary size $|\mathcal{V}|$.

We observe that most of the keywords set do not have a significant gapped intersection size with the query, hence it is highly likely that their sketches share *no* common hash value with the sketch of the query. It is thus desirable to consider only those keywords that share at least one hash value in their appropriate multiset bottom-k sketches with the bottom-k sketch of the query.

Therefore, we propose to build an inverted index, which maps (*hashValue*, *index*) to a keyword v. Intuitively, by probing the inverted index with every hash value in S_Q's sketch, we can obtain a list of candidate keywords. Due to the range decomposition, we also have the additional constraint that the indices of these shared hash values must agree with those calculated for the current δ (i.e., returned by DecomposeRange).

Algorithm 4 gives the pseudocode for the algorithm. Initially the candidate set *Cand* is empty (Line 1). In Line 2, we obtain one or two pairs of indices, indexing into S_Q's and a potential candidate set's multiset bottom-k sketches. We iterate over all the pairs. For each pair of indices, we use i_A to retrieve the sketch of S_Q, and use Line 5–6 to retrieve all keywords such that their sketches with index value i_B share the same hash value (i.e., e in the code). This step is aided by the precomputed inverted index as a simple index lookup. Finally, we perform range estimate between S_Q and the set of each candidate keywords and return the largest K keywords.

Algorithm 4: Top$K$RangeEstimation

Input : S_Q: sets of query positions; δ: query gap; K: top-K; I: the inverted index that maps hash values (and sketch's indices) to a keyword.

Output: Top-K keywords based on their estimated intersection size with S_Q

$R \leftarrow \emptyset$; $Cand \leftarrow \emptyset$;

$arr \leftarrow$ DecomposeRange(δ) ;

for each $(i_A, i_B) \in arr$ **do**

  Retrieve S_Q's multiset bottom-k sketch whose index value is i_A into M_Q;

  for each element $e \in$ elems($msk(M_q)$) **do**

    $Cand \leftarrow Cand \cup I[(e, i_B)]$; `/* A list of keywords will be returned by looking up the index` I `*/`;

for each keyword $v \in Cand$ **do**

  $R \leftarrow R \cup (v,$ RangeQueryOnlineEstimation$(S_Q, S_v, \delta))$;

return the K largest entries in R in terms of the estimated intersection size

The algorithm issues at most $2k-2$ index lookups, and performs $|Cand|$ number of range estimations, where $|Cand|$ is the size of the candidate set. Finding the top-K entries from R and sorting them take $O(K \log K)$ time. Therefore, the total estimation time is $O(|Cand| \cdot k + K \log K)$.

In our implementation, we also perform the following optimizations. For every keyword returned from the index lookup (Line 6), we can accumulate its partial inner product with the query's sketch (i.e., the numerator of Eq. (3.3)). It can be shown that the numerator's value will be correctly calculated after the loop of Lines 3–6 ends. Therefore, the RangeQueryOnlineEstimation function only needs to do $O(1)$ computation to get the range estimation for each candidate.

Table 3.6: Inverted Index for top-K Related Keyword Mining

Key	**List of Keywords**
$(0.15, \{9\})$	S_A
$(0.16, \{3, 6\})$	S_B
$(0.16, \{9\})$	S_A
$(0.21, \{9\})$	S_B
$(0.22, \{1, 2, 3\})$	S_A, S_B
$(0.22, \{2, 3\})$	S_A
$(0.22, \{3, 6\})$	S_A
...	...

Example 3.6. *Using the same example, we can build the inverted index as shown in Table 3.6.*

Consider the top-1 related keyword query for A with $\delta = 5$. DecomposeRange gives us one pair of indices $(\{3, 6\}, \{1, 2, 3\})$. This means we are concerned with the query's sketch indexed by $\{3, 6\}$ and the candidate's sketch indexed by $\{1, 2, 3\}$. We load the query sketch $msk(S_A^{+3,6}) = \{0.22 : 1, 0.42 : 2, 0.48 : 1\}$. We issue three lookups with keys: $(0.22, \{1, 2, 3\})$, $(0.42, \{1, 2, 3\})$, and $(0.48, \{1, 2, 3\})$. These lookups find matches S_B, which is added to the candidate set. Finally we per-

form range estimation between the query and the candidates then return the top-1 keyword.

3.5 Experimental Evaluation

In this section, we present the results of a comprehensive performance study to evaluate the efficiency and effective- ness of the proposed techniques.

3.5.1 Experiment Setup

We use the following algorithms for point and range estimation.

- GSISE-k is our proposed point (range) estimation method and k determines the size of a single bottom-k sketch.
- PointSum-k is our basic range estimation methods by summing up multiple point estimation results.
- Exact calculates the exact answer for point and range estimation, respectively.

We use the following algorithms for the top-K related keyword query.

- TopK is the hash table based top-K range estimation method proposed in Section 3.4.
- $\text{Exact}_{\text{topk}}$ is the exact algorithm for top-K range estimation. The algorithm is conducted by scanning the whole dataset, finding each occurrence of the query keyword, and bookkeeping the count of every other keyword that occurs within the δ neighborhood. Finally, it returns the K keywords with highest number of occurrences.

The dataset we use is a subset of the **ClueWeb** containing 500 million English web pages from the ClueWeb09 collection.[3] We remove infrequent keywords and keep the 100k most frequent keywords. We build the positional inverted index for these keywords, and treat each inverted list as a set. We then build three sets of sketches on these sets, i.e., point sketch, range sketch, and top-K sketch for the respective estimation problems.

The complete original positional index requires more than 2 days to construct using Hadoop on a cluster of 20 PCs, and the final index is stored on the HDFS of the Hadoop cluster. Without compression, the overall index consumes around 1TB space, which is impossible to load into a single commodity PC's memory. For point sketch, the index size is 160GB. For range sketch and topK sketch, the index size is 240GB, respectively. Thus the sketch is small enough to fit into our testing environment with 256GB RAM. Therefore, the experiments conducted on the sketches are memory-based, while exact algorithms that processes inverted lists or scanning documents have to use disk I/Os.

Estimation Workload. We randomly select 100 pairs of keywords to perform point and range estimation with different parameter settings on their corresponding sets. We set maximum gap $N = 100$. k varies between 1,000 to 100,000 (default). The query gap δ varies between 20 and 100.

We measure the **running time** and **relative error** for point and range estimation methods. We measure **recall** and **extended recall** for TopK methods. Recall is defined as A_K/K, where A_K is the number of exact top-K results returned by an algorithm that outputs K results. The extended recall is defined by $A_{L\times K}/K$, where $A_{L\times K}$ is the number of exact top-K results returned by an algorithm that outputs $L \times K$ results, L is the extension rate. All measurements shown are averaged over

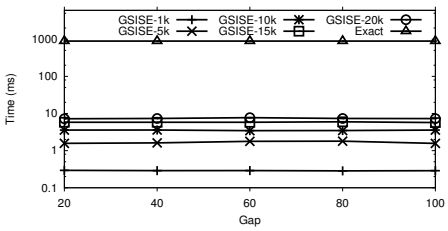
[3] `http://lemurproject.org/clueweb09.php`

100 queries.

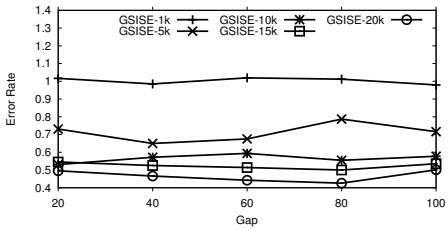
The experiment parameter settings of the evaluated algorithms are listed in Table 3.7, where the default parameters are highlighted in bold.

Table 3.7: Parameter Settings

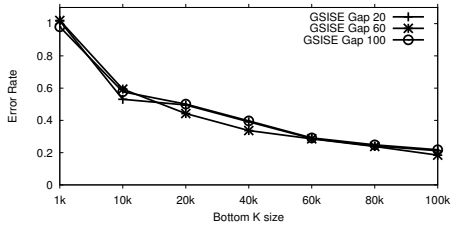
Parameters	Setting (Defaults are in bold)
maximum gap	**100**
top-K	**1**, 10, 100
bottom-k	1000, 5000, 10000, 15000, **20000**
query gaps	20, 40, 60, 80, **100**



(a) Point Estimation Time



(b) Point Estimation Relative Error



(c) Point Estimation Relative Error

Figure 3.2: Experiment Results of Point Query

3.5.2 Point Estimation

In Figure 3.2(a), we show the point estimation time by varying query gaps. Compared with the exact algorithm, the sketch based method is more than 2 order of magnitudes faster. Query gaps do not affect the estimation time and exact time.

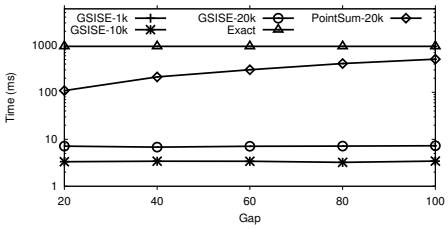
As both exact and estimation methods rely on intersection algorithm of complexity $O(k)$ to find common elements. GSISE-1k, GSISE-5k, GSISE-10k, GSISE-15k, and GSISE-20k spend 0.38, 1.66, 3.47, 5.03 and 6.92 milliseconds to perform one estimation. The trend is linear. Average length of inverted list size is around 2.5 million, which is 125 times of GSISE-20k's sketch size. Meanwhile, Exact algorithm uses $892ms$ to calculate exact answer, which is $892/6.92 = 128$ times of GSISE-20k's estimation time. When dataset size increases, exact results will need longer time, while the sketch estimation time is immune to the increasing dataset size.

In Figure 3.2(b) and figure 3.2(c), we show how the relative error decreases while the bottom-k sketch size is increasing. When gap is 80, the relative error drops from 1 to 0.4 with k changing from 1000 to $20,000$. By further increasing k up to $100,000$, the relative error reduces to less than 0.19. Based on the analysis in section 3.2.2, the relative error is inverse proportional to the square root of bottom-k size. Despite the fluctuation, relative error is not affected by the different query gaps, the fluctuation is caused by different intersection sizes at different gaps. In our experiments, gap 40 has slightly more intersections than gap 100, which reflects that gap 40 has slightly less relative error than gap 100.

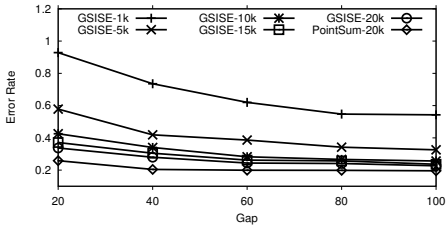
3.5.3 Range Estimation

From Figure 3.3(a) to 3.3(c), we present range query estimation results.

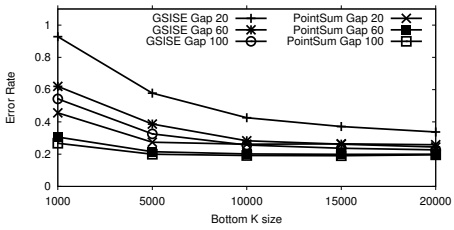
- **Time.** Figure 3.3(a) shows that our GSISE range estimation method has the best performance compared to other methods. For example, GSISE-20k is 128 times faster than Exact and 108 times faster than PointSum-20k when gap is 100. Similar to point estimation, GSISE is stable with different gaps. According to Algorithm 2, any range query can be decomposed into at most two estimation queries, and the efficiency of estimation only relies on k. PointSum



(a) Range Estimation Time



(b) Range Estimation Relative Error



(c) Range Estimation Relative Error

Figure 3.3: Experiment Results of Range Query

performs individual point estimations for each gap and uses the summation as the estimation result. Thus the processing time of PointSum is linear to the query gap. When gap is 20, 40, 60, 80 and 100, the processing time of PointSum-20k is 109.0, 213.1, 302.7, 410.6, and 508.0 milliseconds, which is linearly increasing with respect to gap.

- **Relative Error.** As shown in Figure 3.3(b), when query gap increases, the relative errors of all methods decrease. This is because larger range gap tends to result in larger intersection size. According to Theorem 3.3, it is equivalent to using a larger k.

  It is noted that PointSum achieves slightly lower relative error than range estimation given the same k setting. The reason is that PointSum utilizes all values in the point shifted sketch for estimating, which is equivalent to merging *without* truncating at k-th hash value. As opposed to the truncated merge operation applied to GSISE_r index, PointSum will consume up to more

than λ times space.To be more specific, in Figure 3.3(b), when query gap is 100, the range sketch required for GSISE-20k is $2 \times 20k$, while for PointSum, the cost is $20 \times 20k$. However, the PointSum-20k only decrease the error by less than 0.05. Considering the tremendous performance gain in efficiency and storage, the tiny sacrifice in relative error can be ignored.

Figure 3.3(c) shows similar trend as Figure 3.2(c). Larger k will lead to smaller relative error. The relative error decreases rapidly at the initial increasing of k, then flattens after $k > 10,000$. PointSum has smaller relative error than GSISE methods, however, the relative errors are almost the same when k is large. Interestingly, the storage cost of PointSum-1k of gap $100(20 \times 1k = 20k)$ and of GSISE-10k of gap $100(2 \times 10k = 20k)$ are the same, so are the relative error of PointSum-1k of gap 100 and GSISE-10k, which are 0.267 and 0.266, respectively.

3.5.4 Top-K Related Keywords Mining

The experiment results of top-K problem are presented in Figures 3.4(a) to 3.4(i).

- **Time**. In Figures 3.4(a), 3.4(d) and 3.4(g), we investigate the response time by varying K of top-K from 1 to 100. In general, we can see that the average query processing time is under 25 milliseconds, which is quite small compared to $\mathsf{Exact}_{\mathsf{topk}}$. In fact, as we use large web page corpus, the $\mathsf{Exact}_{\mathsf{topk}}$ algorithm must run on the cluster in a batch mode for the given queries, which takes 6 hours to scan through the original documents in order to output top-K results. In each figure, by increasing the query gaps from 20 to 100, we observe that the query processing time increases slightly. This is due to the processing of top-K range estimation needs to count the number of common sketch values

in the hash table. Larger query gap will result in larger intersection size, which means more common sketch values when looking up the hash table. Same reason goes for the increasing query processing time when k increases.

- **Recall**. We show the recall of our algorithms in different top-K settings in Figures 3.4(b), 3.4(e) and 3.4(h). We can observe that the recall grows with the increase of k. Larger k incurs smaller relative error, consequently improves the recall of the top-K algorithms. Furthermore, it is observed that query gap has no influence on the recall, which shows great stability over different query gap settings.

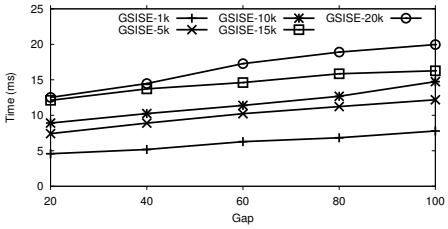
- **Extended Recall**. As shown in Figure 3.4(a) and Figure 3.4(g), the processing time of returning top 1 result and top 100 results are similar. This gives us the motivation of measuring extended recall. Figures 3.4(c), 3.4(f) and 3.4(i), present the extended recall. The extension rate is up to 4. In Figure 3.4(c), when extension rate is larger than 3, the recall of GSISE-15k and GSISE-20k both reach 90%. In other figures, all GSISE-20k experiments can reach more than 80% of recall when extension rate is 4. GSISE-1k performs inadequate even when extension rate reaches 4.

  In summary, our top-K estimation algorithm shows substantial performance advantage over exact methods in efficiency. Meanwhile, the recall of our algorithms can reach around 90% with slight extension of returned top-K results.

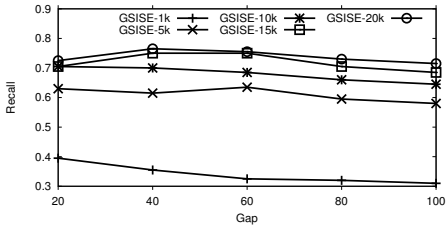
3.6 Conclusions

In this chapter we formally define the GSISE problem, for both point and range gap constraints. We propose space and time efficient estimation methods, based

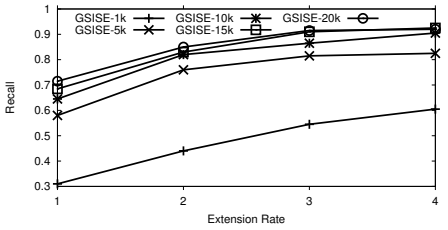
on bottom-k sketches and its extension to multisets. In addition, our estimation methods provide the probabilistic quality guarantees. We also apply our technique to the problem of finding top-K related keyword, by combining our estimation technique with the use of an inverted index. Our experiments using half a billion documents empirically verify the effectiveness and efficiency of the proposed methods.



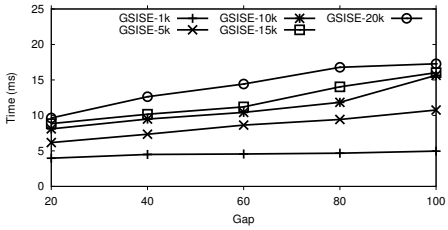
(a) Top 1 Query Time



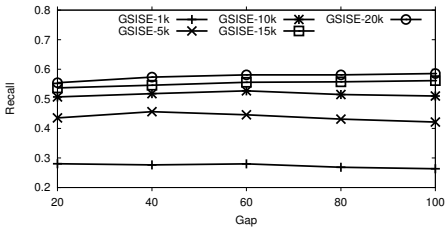
(b) Top 1 Query Recall



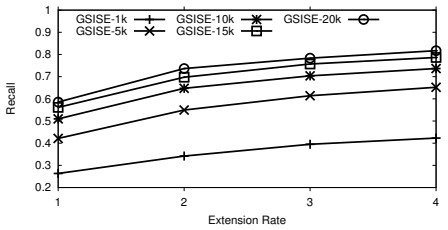
(c) Top 1 Query Extended Recall



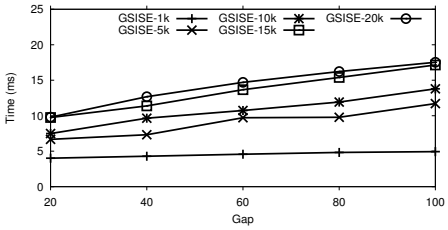
(d) Top 10 Query Time



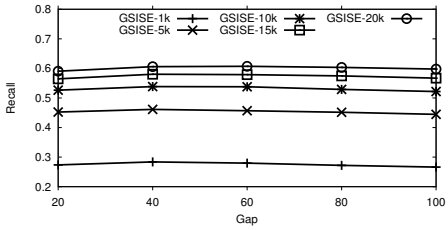
(e) Top 10 Query Recall



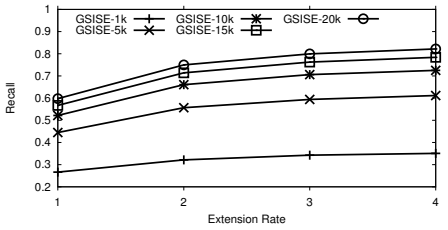
(f) Top 10 Query Extended Recall



(g) Top 100 Query Time



(h) Top 100 Query Recall



(i) Top 100 Query Extended Recall

Figure 3.4: Experiment Results of Top-K Query

Chapter 4

Effective Order Preserving Estimation Method

4.1 Overview

Statistical method is essential for analyzing the massive data that arises in many applications. A synopsis of a large dataset captures vital properties of the original data while typically consuming much less resources. One of the most widely used properties is the population mean, which is the average value of a group of numeric data. The methods and criteria for a reliable population mean estimation are well developed in both statistical inference and computer science areas. However, when it comes to several groups of data, it is long neglected that an order estimation on the population means is of equal importance, since many applications require to know data trends or more specifically the relationships among variables [CGHJ12, SWQ+14, IBS08]. Motivated by above observations, we study the order preserving estimation (OPE) problem in this chapter. Given k groups of numeric data with unknown distribution along with $\delta \in (0, 1)$, OPE returns an order estimate on the

group average with a probabilistic guarantee $1 - \delta$. With the rapid development of modern computing ability, the offline sample processing time is less concerned in many applications. Instead, the sample size becomes a critical factor to be restrained. For example, the I/O cost of obtaining data from external or distributed storage is very high. Or in most clinical trials of new drugs, the number of human subjects is usually limited due to the risk or ethical issues. In both cases, it is not affordable to get as many samples as we want. Therefore, we consider the total sample size as the most important parameter in our algorithm design.

The state-of-the-art method [KBP+15] guarantees to output a correct order and is proved to be near optimal in the total sample size. Nevertheless, an important assumption for the near-optimality is that it does not allow the inactive confidence intervals to become to be active again. The assumption affects the logical strengths of the proof. Because the crude elimination of sampling outliers is unrealistic in real applications. Even though the assumption could be ignored without impairing the theoretical soundness, [KBP+15] tends to examine more samples than needed, which means their sample complexity upper bound is conservative empirically.

Contribution. Our improvements have two aspects, stop condition and sample strategy, respectively. We design two stop functions, named IntervalSeparation and PairwiseComparison. They utilize the relation among current sample means to make judicious decisions, in order to output a correct result while using as little samples as possible. We also propose a heuristic sample strategy to assign new sample points adaptively. Specifically, IntervalSeparation reduces order estimates to separating the underlying confidence intervals. Due to tail inequality, all true means can be bounded within their confidence intervals with high probabilites. Thus as long as the intervals do not overlap, the order can be guaranteed. Next in the PairwiseComparison method, we first show that minimizing the sample size in OPE

is NP-hard by reducing from TSP problem on tournament graphs. In order to compute the failure probability, we introduce the prefix downsample technique to equalize the sample size between adjacent sample groups, where the unequal sample size is due to the proposd sample strategy.

The rest of the chapter is organized as follows: Section 4.2 defines the problem and gives an overview of our algorithm framework. Section 4.3 introduces the interval separation algorithm and proves its correctness. Section 4.4 analyzes the pairwise comparison algorithm and the prefix downsample technique, along with the sample strategy. Section 4.5 shows our experimental results and Section 4.6 concludes the chapter.

4.2 Background

We first formulate the problem of order preserving estimation, then give an overview of our proposed framework.

4.2.1 Problem Definition

An important concept used in our algorithms is the confidence interval. Given $\epsilon, \delta \in (0, 1)$, for an estimated sample mean $\hat{\mu}$, let its confidence interval be $[\hat{\mu}-\epsilon, \hat{\mu}+\epsilon]$ with confidence level $1-\delta$. It means that the population mean μ belongs to $[\hat{\mu}-\epsilon, \hat{\mu}+\epsilon]$ with at least $1-\delta$ probability, which is denoted by $\mathbf{Pr}\left[|\mu-\hat{\mu}| \leq \epsilon\right] \geq 1-\delta$.

In this chapter, we use the empirical Bernstein-Serfling inequality proposed in [BM15] to derive robust and tight confidence intervals.

Definition 4.1 (Order Preserving Estimation)**.** *Given k groups $g_1, g_2, \cdots, g_k$ of numeric values where the values in g_i are bounded by $[a_i, b_i]$. Denote the group population mean of elements in g_i as $\mu_i = \frac{1}{|g_i|}\sum_{e \in g_i} e$. Given $\delta \in (0, 1)$, an order*

preserving estimation for the group mean returns an order estimate on μ_i *for* $i \in [1, k]$*, s.t. the order is correct with at least* $1 - \delta$ *probability. Correct order means if the population means satisfy* $\mu_i > \mu_j$*, the sample means also have* $\hat{\mu}_i > \hat{\mu}_j$ *for all* $i, j \in [1, k]$.

As justified in Section 1.1, the sample size is a major concern in various applications. In order to meet the ultimate user requirements, we model OPE as an minimization problem. The goal is to minimize the sample complexity. In such case, we consider the total sample size as the dominant factor in our analysis.

Definition 4.2 (Minimization Problem for OPE)**.** *Let* n_i *be the number of samples taken from group* i *before the algorithm terminates, OPE aims to minimize* $\sum_{i=1}^{k} n_i$*, which is the total sample complexity of all the* k *groups.*

To achieve the minimization goal, we need to relate the sample complexity $\sum_{i=1}^{k} n_i$ to the user specified failure probability δ in the randomization framework. Therefore we adopt the empirical Bernstein-Serfling inequality for sampling without replacement [BM15], along with the union bound as our primary tools to solve the problem. The superiority lies in the fact that they does not make assumptions on the data distribution. As a consequence, OPE can return a correct order for *any* data distribution with a high probability.

4.2.2 Framework of OPE

In this section, we give an overview of our algorithms, followed by the intuition of our improvements.

The user issues a query about the order of k groups. An efficient **sample strategy** is designed to incrementally get random samples and update sample means. We use random sample without replacement throughout the chapter, which

will lead us to the population mean when the sample size equals population size. In the meantime, a **stop function** computes an order estimate and decides if it satisfies the probabilistic guarantee hence can stop. Above process is repeated until the stop condition is satisfied. The procedure is summarized in Algorithm 5.

Algorithm 5: OPE

Input: Data from $g_1, g_2, \cdots, g_k$, δ.

Output: Order estimates $\hat{\mu}_1, \hat{\mu}_2, \cdots, \hat{\mu}_k$ on the group means.

```
1 n ← 1;
2 Draw n samples from g_i to produce initial sample mean μ̂_i, i ∈ [1, k];
3 F ← StopFunction;          /* Estimate an order and compute the stop
  function */;
4 while F > δ do
5     Get new samples according to SampleStrategy then update μ̂_i;
6     F ← StopFunction;          /* Update order and stop function */;
7 return Estimated order
```

Our key observation is that some groups are easy to be ordered correctly, as their population means have large difference. We call it easy case which only needs few samples to get a correct order. If relating easy case to the stop function, it will result in a small failure probability δ_i as the order estimate is unlikely to be incorrect. On the contrary, hard case refers to ordering the groups with small differences among their population means. It will cause large δ_i hence consuming a large amount of samples for a correct result. Therefore, we need to design a judicious sample strategy that can assign random samples *adaptively*, as well as effective stop functions that can allocate δ_i *dynamically*. We will materialize StopFunction in Algorithm 5 as IntervalSeparation (Section 4.3) and PairwiseComparison

(Section 4.4). SampleStrategy is introduced at the end of Section 4.4. Table 4.1 lists the notations frequently used throughout the chapter.

Table 4.1: Summary of Notations

Symbol	Explanation
g	group to be ordered
k	number of groups
μ	population mean
$\hat{\mu}$	sample mean
ϵ	error parameter of the confidence interval
δ	failure probability of the confidence interval
N	population size
n	sample size
a	minimum of a given group
b	maximum of a given group

4.3 Interval Separation Method

We first give the intuition of IntervalSeparation, then define the stop function in Section 4.3.1 followed by its proof of correctness in Section 4.3.2.

Intuition. To order group g_i and g_j, we can examine the confidence intervals of their population means μ_i and μ_j. If both μ_i and μ_j are within respective confidence intervals with high probabilities, and the two intervals do not overlap, we can output an order that is probabilistic correct. When it comes to k groups, the idea is interleaving the interval boundaries $\{x_1, \cdots, x_{k-1}\}$ with k sample means that are ordered decreasingly, then deriving the total failure probability based on the union bound.

4.3.1 IntervalSeparation Stop Function

We will order the sample means decreasingly before evaluating the stop function $f_{\min}$. The ordered means are denoted by $\hat{\mu}_1, \cdots, \hat{\mu}_k$. To keep the notation clean, we use $\hat{\mu}_i$ as the i-th largest sample mean in the rest of Section 4.3, instead of the sample mean of i-th group used previously. As shown in Equation (4.1), we first define function f, then minimize it in terms of $\mathbf{x} = \{x_1, \cdots, x_{k-1}\}$, where $x_i \in (\hat{\mu}_{i+1}, \hat{\mu}_i)$, and use $f_{\min}$ as the stop function. If $f_{\min}$ is no greater than the user specified δ, the algorithm are safe to stop. $f_{\min}$ in Equation (4.1) will replace Line 3 and Line 6 of Algorithm 5. The stop condition is to decide if $f_{\min}$ exceeds δ at each round. Determining $f_{\min}$ is equivalent to finding the root of $\frac{\partial f}{\partial x_i}$ between $x_i \in (\hat{\mu}_{i+1}, \hat{\mu}_i)$. Since the closed-form expression for the root is unavailable, we use binary search to get an approximate result.

$$
\begin{aligned}
f(\mathbf{x}) = 5 \sum_{i=1}^{k-1} \Bigg[& \exp\left(- \left(\frac{\hat{\sigma}_i \sqrt{2\rho_i n_i} + \sqrt{2n_i(\rho_i \hat{\sigma}_i^2 + 2r_i \kappa(\hat{\mu}_i - x_i))}}{2r_i \kappa} \right)^2 \right) \\
& + \exp\left(- \left(\frac{\hat{\sigma}_{i+1} \sqrt{2\rho_{i+1} n_{i+1}} + \sqrt{2n_{i+1}(\rho_{i+1} \hat{\sigma}_{i+1}^2 + 2r_{i+1} \kappa(x_i - \hat{\mu}_{i+1}))}}{2r_{i+1} \kappa} \right)^2 \right) \Bigg], \\
f_{\min} = \min & f(\mathbf{x}), \text{ subject to } x_i \in (\hat{\mu}_{i+1}, \hat{\mu}_i),
\end{aligned}
\tag{4.1}
$$

where for each group g_i, $\hat{\sigma}_i$ is the sample standard deviation, $\hat{\mu}_i$ is the sample mean, n_i and N_i are the sample and population size, $r_i = b_i - a_i$, $\kappa = \frac{7}{3} + \frac{3}{\sqrt{2}}$, and:

$$
\rho_i = \begin{cases} 1 - \frac{n_i - 1}{N_i} & , \text{if } n_i \leq \frac{N_i}{2}. \\ (1 - \frac{n_i}{N_i})(1 + \frac{1}{n_i}) & , \text{if } n_i > \frac{N_i}{2}. \end{cases}
$$

4.3.2 Analysis

We prove the correctness of IntervalSeparation in Theorem 4.1.

Theorem 4.1. *Replacing Line 3 and Line 6 of Algorithm 5 with Equation* (4.1) *can return a correct order with at least* $1-\delta$ *probability.*

Proof. Given a group of size N, assume each sample X_i is bounded by $[a, b]$, then we know the sample mean $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} X_i$, and the population mean $\mu = \mathbf{E}[\hat{\mu}]$. Given ordered sample means $\hat{\mu}_1, \cdots, \hat{\mu}_k$, let $x_i \in (\hat{\mu}_{i+1}, \hat{\mu}_i)$, and define $\epsilon_i = |x_i - \hat{\mu}_i|$. According to the empirical Bernstein-Serfling Inequality [BM15], for the upper side of g_i, we have

$$\mathbf{Pr}[\mu_i \leq x_i] \leq 5\exp\left(-\left(\frac{\hat{\sigma}_i\sqrt{2\rho_i n_i} + \sqrt{2n_i(\rho_i\hat{\sigma}_i^2 + 2r_i(\hat{\mu}_i - x_i)\kappa)}}{2r_i\kappa}\right)^2\right).$$

Analogously, for the lower side of g_{i+1},

$$\mathbf{Pr}[\mu_{i+1} \geq x_i] \leq 5\exp\left(-\left(\frac{\hat{\sigma}_{i+1}\sqrt{2\rho_{i+1} n_{i+1}} + \sqrt{2n_{i+1}(\rho_{i+1}\hat{\sigma}_{i+1}^2 + 2r_{i+1}(x_i - \hat{\mu}_{i+1})\kappa)}}{2r_{i+1}\kappa}\right)^2\right).$$

Above derivations are applicable to all the boundaries x_i where i belongs to $[1, k-1]$. Due to the continuity of f and the union bound, the correctness is concluded by

$$\mathbf{Pr}\left[\bigvee_{i=1}^{k-1}\left(x_i \in (\hat{\mu}_{i+1}, \hat{\mu}_i)\right)\right] \geq 1 - f_{\min} \geq 1 - \delta.$$

□

Remember that a confidence interval has two ingredients ϵ and δ. It can be interpreted as $\mathbf{Pr}[\text{error is within } \epsilon] \geq 1-\delta$. Given k groups, the methods in [KBP+15] fix all the δ_i to be $\frac{\delta}{k}$, and derive the associated interval $[\hat{\mu}_i - \epsilon_i, \hat{\mu}_i + \epsilon_i]$ then test if there are overlaps among different intervals. The difference between IntervalSeparation and that in [KBP+15] is that we do it inversely. To be specific, we first place $\mathbf{x}$ as interval boundaries that satisfy the nonoverlapping condition. Then we compute the associated δ_i and define $\sum_{i=1}^{k-1}\delta_i$ as $f(\mathbf{x})$. Next we test if there exists

$\mathbf{x}$ such that $f_{\min}$ is no greater than the given δ. It is equivalent to allocating δ_i dynamically. The advantage is that it assigns less δ_i to the easy cases while more δ_i to the hard cases for an early termination.

4.4 Pairwise Comparison Method

We first give the intuition of PairwiseComparison. Then we illustrate the steps for doing prefix downsample between adjacent groups with different sample sizes and explain its necessity. After that we define the stop function and prove it is NP-hard to solve the OPE minimization problem in Definition 4.2.

Intuition. Suppose we have values μ_a, μ_b and μ_c and want to order them decreasingly. A natural way is to show $\mu_a - \mu_b > 0$ and $\mu_b - \mu_c > 0$. Then it must have $\mu_a > \mu_b > \mu_c$ according to the transitivity of the greater than relation. We apply this simple yet effective idea with probabilistic constraint and propose the stop function for PairwiseComparison. At first glance, the idea may seem straightforward to apply. However, there are several issues if we would like a theoretical guarantee while making the sample size as small as possible.

- We need to determine an order as the input to the stop function.
- The complexity of finding the order with minimum output value is NP-hard.

4.4.1 Prefix Downsample

This section considers prefix downsample method that can be maintained efficiently as the samples update. The sampling method we use is random sample without replacement.

Definition 4.3 (Prefix Downsample)**.** *Given a group, denote its sample as an ordered*[1] *set* $S = \{X_1, X_2, \cdots, X_n\}$*, where* n *is the sample size. A prefix downsample of size* m *is defined as* $S_{pre} = \{X_1, X_2, \cdots, X_m\}$*, which is the first* m *elements of* S*.*

Given S and the corresponding S_{pre}, *prefix downsample mean* is the sample mean of m samples in S_{pre}, while full sample mean refers to the standard sample mean of S. It is easy to see that both of them are unbiased estimators of the population mean. Lemma 4.1 shows that a prefix downsample of S is a correct random sample for evaluating the stop function.

Lemma 4.1. *Given a random sample* S*, any prefix* S_{pre} *of size* m *is a valid prefix downsample that can be used for evaluating the stop function.*

The proof is immediate by showing S_{pre} is also a valid random sample for the same population. Let population size be N, sample size $|S|$ be n, downsample size $|S_{pre}|$ be m. Then the probability of an element e appearing in the downsample is

$$\mathbf{Pr}\left[e \in S_{pre}\right] = \frac{n}{N} \cdot \frac{m}{n} = \frac{m}{N},$$

which is equal to the probability of sampling m points directly from the population. As a result, any prefix of size $m < n$ is also a random sample of the same population hence can be used to evaluate the stop function. Next we consider how to apply prefix downsample between two groups and define the relation "$>_{pre}$" between them.

Definition 4.4 (Binary Relation "$>_{pre}$" and "$-_{pre}$")**.** *Given two groups* g_i *and* g_j *with sample size* n_i *and* n_j*, denote their full sample means as* $\hat{\mu}_i$ *and* $\hat{\mu}_j$*. Let the*

[1]The order is induced from progressive sampling without replacement process. I.e., $Rank(X_i) < Rank(X_j)$, if $i < j$.

prefix downsample size m *be* $\min(n_i, n_j)$ *for both groups. Then it holds* $\hat{\mu}_i >_{pre} \hat{\mu}_j$ *if the prefix downsample mean of* g_i *is greater than that of* g_j*. In addition, we use* $\hat{\mu}_i -_{pre} \hat{\mu}_j$ *to denote the difference between the prefix downsample means of* g_i *and* g_j*.*

Prefix downsample is designed to equalize the sample size between adjacent groups to be compared in Equation (4.4), in order to apply the empirical Bernstein-Serfling inequality. To be more specific, assume the adjacent groups are g_i and g_j. We will define $\hat{\mu}_i -_{pre} \hat{\mu}_j$ as an *individual* random variable to apply the inequality. It means the sample size of g_i and g_j must be identical due to the requirement of empirical Bernstein-Serfling inequality. Therefore, we need to find out the criteria for extracting a correct downsample that makes the most of available samples and can be maintained efficiently as S updates. Although we can use any size m prefix in terms of the algorithm correctness, we deem $m = \min(n_i, n_j)$ as the most effective one. Since it manages to take advantage of all the sample information in the adjacent groups with sample size n_i and n_j, respectively.

4.4.2 PairwiseComparison Stop Function

Before introducing the stop function h, we need to define candidate order as the input to the function.

Remember in the IntervalSeparation section, we need to arrange all the sample means as an ordered set as the input to the stop function. However, in the PairwiseComparison, the procedure can not be finished directly. We name the order by *candidate order*, and show that finding the candidate order with smallest stop function h value is NP-hard.

Candidate Order. Given k groups and an arbitrary order $g_1, g_2, \cdots, g_k$, [2] g_i

[2]Same as before we use g_i as the group with rank i in a candidate order in the rest of

denotes the i-th rank in the order, and adjacent groups refer to (g_i, g_{i+1}), $i \in [1, k-1]$. If it holds $\hat{\mu}_i >_{pre} \hat{\mu}_{i+1}$ for all $i \in [1, k-1]$, we say that $g_1, g_2, \cdots, g_k$ is a candidate order.

We prove in Lemma 4.2 the existence of a candidate order.

Lemma 4.2. *There exists at least one candidate order in OPE.*

Proof. We finish the proof by mathematical induction. Let the number of groups $n = 2$, due to prefix downsample, we can calculate two sample means $\hat{\mu}_1$ and $\hat{\mu}_2$. Suppose $\hat{\mu}_1 >_{pre} \hat{\mu}_2$, then the candidate order is g_1, g_2. Thus for $n = 2$ the statement is true.

Assume for $n = k$, there exists a candidate order $g_1, g_2, \cdots, g_k$, which is

$$\hat{\mu}_1 >_{pre} \hat{\mu}_2 >_{pre} \cdots >_{pre} \hat{\mu}_k.$$

In the induction step, we exhaust all the possible relations between $\hat{\mu}_{k+1}$ and the candidate order $g_1, g_2, \cdots, g_k$. Then we show that there must exist a construction for a valid candidate order in all the relations.

For $n = k+1$, if $\hat{\mu}_{k+1} >_{pre} \hat{\mu}_1$, we can form a candidate order $g_{k+1}, g_1, g_2, \cdots, g_k$.

If $\hat{\mu}_k >_{pre} \hat{\mu}_{k+1}$, we can form a candidate order $g_1, g_2, \cdots, g_k, g_{k+1}$.

We now consider the case when $\hat{\mu}_1 >_{pre} \hat{\mu}_{k+1}$ and $\hat{\mu}_{k+1} >_{pre} \hat{\mu}_k$. If $\hat{\mu}_{k+1} >_{pre} \hat{\mu}_2$, we can form a candidate order $g_1, g_{k+1}, g_2, \cdots, g_k$.

When $\hat{\mu}_2 >_{pre} \hat{\mu}_{k+1}$, if $\hat{\mu}_{k+1} >_{pre} \hat{\mu}_3$, we can form a candidate order $g_1, g_2, g_{k+1}, g_3, \cdots, g_k$.

Repeat the process until we assume $\hat{\mu}_i >_{pre} \hat{\mu}_{k+1}$, where $i \in [1, k-1]$. As we know $\hat{\mu}_{k+1} >_{pre} \hat{\mu}_k$, then a candidate order is formed by $g_1, g_2, \cdots, g_{k-1}, g_{k+1}, g_k$.

Therefore the statement also holds when $n = k + 1$. To conclude, there exists at least one candidate order in OPE. □

Section 4.4 in order to keep the notation clean.

Now suppose we have a candidate order with full sample means $\hat{\mu}_1, \cdots, \hat{\mu}_k$, we can define the stop function h for PairwiseComparison in Equation (4.2) . h will replace Line 3 and Line 6 of Algorithm 5. The stop condition is to decide if h exceeds δ at each round.

$$h = \sum_{i=1}^{k-1} 5 \exp\left(- \left(\frac{\hat{\sigma}_i \sqrt{2 n_i \rho_i} + \sqrt{2 n_i \left(\rho_i \hat{\sigma}_i^2 + 2 r_i \kappa |\hat{\mu}_i -_{pre} \hat{\mu}_{i+1}| \right)}}{2 r_i \kappa} \right)^2 \right), \tag{4.2}$$

where for each group g_i, we use $\hat{\mu}_i$ as the sample mean, n_i and N_i as the sample and population size, and:

$$\rho_i = \begin{cases} 1 - \frac{n_i - 1}{\min(N_i, N_{i+1})} & \text{, if } n_i \leq \frac{\min(N_i, N_{i+1})}{2}. \\ (1 - \frac{n_i}{\min(N_i, N_{i+1})})(1 + \frac{1}{n_i}) & \text{, if } n_i > \frac{\min(N_i, N_{i+1})}{2}. \end{cases} \tag{4.3}$$

$r_i = (b_i + b_{i+1}) - (a_i + a_{i+1})$, $\kappa = \frac{7}{3} + \frac{3}{\sqrt{2}}$, and $\hat{\sigma}_i$ is the sample standard deviation of $\hat{\mu}_i -_{pre} \hat{\mu}_{i+1}$.

Our observation is that determining whether the algorithm can stop is equivalent to deciding if there exists a candidate order with $h \leq \delta$. Nevertheless, it is NP-hard to solve the decision problem. It is because a given group g_i may have different prefix downsample means when downsampling with g_{i-1} and g_{i+1}, due to different downsample size $\min(n_{i-1}, n_i)$ and $\min(n_i, n_{i+1})$.

Given sample set S_i, full sample size n_i, full sample mean $\hat{\mu}_i$ and δ, the desicion problem for PairwiseComparison in OPE is defined as $\{\langle (S_i, n_i, \hat{\mu}_i)_{i=1}^k, \delta \rangle$: There exists a candidate order on the k groups, such that it holds $h \leq \delta\}$.

Theorem 4.2. *PairwiseComparison in OPE is NP-hard.*

Proof Sketch. We will reduce from a known NP-hard problem "Traveling-Salesman Problem on Tournament graph (TSPT)"to prove the NP-hardness of OPE. The reduction can be summarized as follows. Without loss of generality, let n_i be arbitrary integers satisfying $n_1 < n_2 < \cdots < n_k$. For each vertex V_i in a tournament

graph G, the reduction algorithm transforms it to group g_i in OPE. Then the cost function c_{ij} between V_i and V_j in TSPT is normalized and equated to Equation (4.5). The sample set S_i hence $\hat{\mu}_i$ can be easily constructed from the equation. Finally the directed edge e_{ij} is transformed to "$>_{pre}$" relation between $\hat{\mu}_i$ and $\hat{\mu}_j$. This process can be done in polynomial time. □

Consequently, we can use TSP algorithms to accelerate the process of finding the candidate order required.

4.4.3 Analysis

We prove the correctness of PairwiseComparison in Theorem 4.3.

Theorem 4.3. *Replacing Line 3 and Line 6 of Algorithm 5 with Equation* (4.2) *can return a correct order with at least* $1-\delta$ *probability.*

Proof. Given a candidate order with full sample mean $\hat{\mu}_i$ for $i \in [1,k]$, we examine the confidence interval of $\hat{\mu}_i -_{pre} \hat{\mu}_{i+1}$. According to the empirical Bernstein-Serfling inequality [BM15], given $\epsilon_i \in [0, \hat{\mu}_i -_{pre} \hat{\mu}_{i+1}]$, it has

$$\mathbf{Pr}\left[\mu_i - \mu_{i+1} > \hat{\mu}_i -_{pre} \hat{\mu}_{i+1} - \epsilon_i\right] \geq 1 - \delta_i. \tag{4.4}$$

The expression for δ_i is

$$\delta_i = 5\exp\left(-\left(\frac{\hat{\sigma}_i\sqrt{2n_i\rho_i} + \sqrt{2n_i\left(\rho_i\hat{\sigma}_i^2 + 2r_i\epsilon_i\kappa\right)}}{2r_i\kappa}\right)^2\right). \tag{4.5}$$

When ϵ_i decreases from $\hat{\mu}_i -_{pre} \hat{\mu}_{i+1}$ to 0, δ_i will increase monotonically. In order to achieve an early stop, we will let ϵ_i be $\hat{\mu}_i -_{pre} \hat{\mu}_{i+1}$. Fit into Equation (4.4), we know

$$\mathbf{Pr}\left[(\mu_i - \mu_{i+1} > \hat{\mu}_i -_{pre} \hat{\mu}_{i+1} - \epsilon_i) \wedge (\epsilon_i = \hat{\mu}_i -_{pre} \hat{\mu}_{i+1})\right] \geq$$
$$1 - 5\exp\left(-\left(\frac{\hat{\sigma}_i\sqrt{2n_i\rho_i} + \sqrt{2n_i\left(\rho_i\hat{\sigma}_i^2 + 2r_i(\hat{\mu}_i -_{pre} \hat{\mu}_{i+1})\kappa\right)}}{2r_i\kappa}\right)^2\right).$$

Apply union bound to all the $k-1$ pairs of adjacent groups hence the correctness is concluded $\square$

Discussion about the proof of correctness. The intuition of our solution is that we would like to dynamically assign the failure probability δ_i when using the union bound, rather than fix each portion to be $\frac{\delta}{k}$. As a consequence, based on the observation $\hat{\mu}_i -_{pre} \hat{\mu}_{i+1} > 0$, we push ϵ_i to the maximum value allowed for concluding that $\mu_i - \mu_{i+1}$ is also greater than 0. One may question the validity of defining the probability of $\mu_i - \mu_{i+1} > 0$, as they are not random variables. However, the randomness originates from Equation (4.4), followed by assigning ϵ_i to be $\hat{\mu}_i -_{pre} \hat{\mu}_{i+1}$. Specifically, $\mathbf{Pr}\left[\mu_i - \mu_{i+1} > 0\right]$ is equivalent to

$$\mathbf{Pr}\left[(\mu_i - \mu_{i+1} > \hat{\mu}_{i+1} -_{pre} \hat{\mu}_{i+1} - \epsilon_i) \wedge (\epsilon_i = \hat{\mu}_i -_{pre} \hat{\mu}_{i+1})\right].$$

Sample Strategy. We introduce our heuristic sample strategy in this section. As the data distribution is unknown to the algorithm, we cannot predict the gain of each sample. Therefore, we adopt the heuristic method to choose the next group to sample from, based on the failure probabilities at current sample round. The intuition is that if the population means of two groups are very close, it will need more samples to separate them than to separate the groups with large difference between their means.

Since our goal is to minimize the sample compelxity instead of the number of rounds the sampling proceeds, we will use the most conservative sampling scheme, i.e. at each round, add one more sample to one of the k groups. Although such method may incur more evaluation cost compared with sampling in a batch mode, it guarantees the algorithm can stop as soon as the success probability is satisfied without consuming unnecessary samples. Specifically, the strategy will randomly choose the next sample from the group that introduces the most failure probability

among all the groups at current round. The proposed sample strategy considers the distribution of all the groups' failure probabilities at current round. Large failure probability implies small difference between sample means, hence it is harder to get the correct order with a high confidence than the groups with far sample means between them. Moreover, the sample means of the groups with small failure probability are stable enough to draw a correct order. Therefore, we spend the rest samples on the groups with large failure probabilities.

4.5 Experiments

In this section, we present the results of a comprehensive performance study to evaluate the effectiveness of the proposed techniques in this chapter.

4.5.1 Experiment Setup

In this chapter we focus on reducing the total sample size required by OPE. Thus we report the sample ratio of the proposed methods by varying different parameters on both real and synthetic datasets.

Algorithms. We compare our methods with state-of-the-art algorithm IFOCUS in [KBP+15]. It stops sampling from a group as long as its confidence interval disjoints with all the other groups. To summarize, the algorithms evaluated in this section are listed below.

- **IFOCUS**. The state-of-the-art approach in [KBP+15].
- **SEP**. The Interval Separation method presented in Section 4.3.
- **COMP**. The Pairwise Comparison method introduced in Section 4.4.

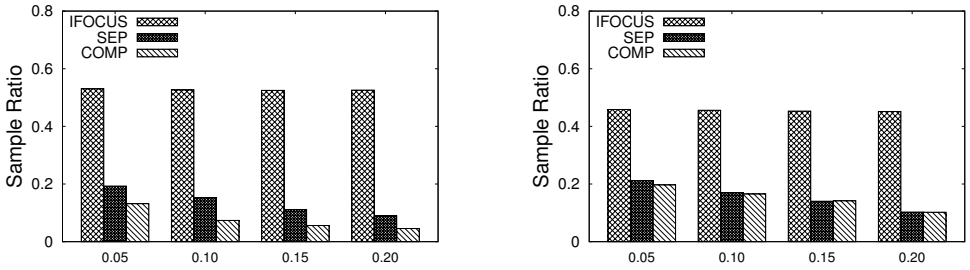
Datasets. We use both real and synthetic datasets. For real datasets, we use the flight records in [Dat09]. We utilize the "ActualElapsedTime" attribute, which denotes the actual travel time for each flight. The attribute represents the marketing preference of the flight company (e.g., short-haul fight or long-haul flight). We use two years' data, 2004 and 2005, which contain 7 million records for each, with 19 and 20 flight companies, respectively. For the synthetic datasets, we generate data from two distributions, uniform (Uniform) and mixture of truncated normal (MixNormal). For each group, we generate 1 million records. For Uniform we randomly select a mean from [0,50] and a data range from [70, 100]. For MixNormal, we select a set of truncated normal distributions in the following manner. Firstly we generate a number from $\{1, 2, 3, 4, 5\}$ indicating the number of truncated normal distributions that comprise the group. For each of the truncated normal distribution, we randomly select a mean from [0,100] and a variance from [1,16].

Workloads. In the experiments, we evaluate the methods by varying the failure probability δ from 0.05 to 0.2 with 0.05 as the default value. The number of groups k increases from 5 to 20 with 20 as the default value. For each setting we run 100 rounds and report the average performance.

Implementation Environment. All experiments are carried out on a PC with Intel Xeon 2.30GHz and 96G RAM. The operating system is Redhat. All algorithms are implemented in C++ and compiled with GCC 4.8.2 with -O3 flag.

4.5.2 Real Data Experiments

In Figure 4.1, we report the sample ratio on the real datasets by varying δ from 0.05 to 0.2. As can be seen, the proposed algorithms, SEP and COMP, require less samples than IFOCUS and can achieve up to 80% reduction in the total sample size. The gain of COMP is more significant on Data 2004 compared with SEP.



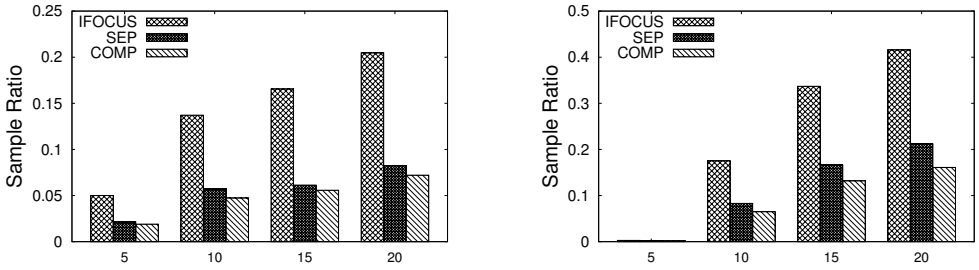
(a) Data 2004 (b) Data 2005

Figure 4.1: Performance Evaluation on Real Datasets by Varying δ

In addition, we notice that IFOCUS is not sensitive to δ. By increasing δ, the sample size does not change much. This is due to the sample size upper bound in the IFOCUS algorithm explained in [KBP+15]. While for our proposed algorithms, increasing δ will make the sample size decrease. This is reasonable as due to the definition of δ, the correctness requirement relaxes with the increase of δ. Hypothetically, any reported order is deemed to be correct when δ equals 1 (fail probability is 100%). If the algorithm is sensitive to δ, the user is able to make a trade off between the sample size and the accuracy by tuning δ. For the correctness of the reported order, IFOCUS always returns the correct order by consuming a large sample size. While for our algorithms, they report wrong orders only when δ equals 0.2. However, the percentage of incorrect orders is less than 3%, which still satisfies the requirement of $\delta = 0.2$ setting.

4.5.3 Synthetic Data Experiments

We will report the performance of the proposed methods on synthetic data by varying the number of data groups k. δ equals 0.05 for all the cases. As shown in Figure 4.2, by increasing k, the sample size increases for all the algorithms. This is because when k increases, we need more samples to bound the correctness of



(a) Uniform Distribution (b) MixNormal Distribution

Figure 4.2: Performance Evaluation on Synthetic Datasets by Varying k

the returned order, which is easy to verify from the stop functions (4.1) and (4.2). IFOCUS increases much faster than the algorithms proposed in this chapter. The sample size needed for Uniform is much smaller than that of MixNormal, since uniform distribution is much easier to infer than normal distribution. COMP requires less samples than SEP, which coincides with the trend on the real datasets. For the accuracy, all the algorithms satisfy the probabilistic guarantee, and IFOCUS always achieves 100% accuracy. This is because IFOCUS consumes more samples than the proposed methods. SEP and COMP have lower accuracy because of the hard cases, which refers to certain groups with very close population means. Due to our dynamic allocation of δ and the sample strategy, we will assign more samples to the hard case. This may lead to the case that we stop sampling from certain groups as long as they satisfy the success probability, even though the order is not correct. It also shows that the proposed algorithms are more sensitive to δ, thus it is able to use less samples while still satisfy the probabilistic guarantee.

4.6 Conclusion

In this chapter we propose two effective stop functions along with a heuristic sample strategy to solve the order preserving estimation problem. In the design of an

effective stop function, we allocate the failure probability δ_i dynamically based on the current observed sample means. Given the total allowed δ, it amounts to allocating small portions of δ to the easy case while save the rest for the hard case. For the sample strategy, we prioritize the hard case to assign more samples, rather than give all the groups the same amount of samples. By conducting empirical evaluations on both synthetic and real datasets, we demonstrate the effectiveness of our proposed methods.

Chapter 5

A Novel Scalable Method for Influence Maximization

5.1 Overview

As a key problem in viral marketing, influence maximization has found many important applications in real life. Given a positive integer k, it aims to find a set of k users in a social network, which can make the largest of adoption or cascade of information. For example, a company wants to choose some influential users to promote its new product. By offering them some free samples or discounts, the company may expect these influential users can propagate the information about the product through their social network, from friends to friends, and finally lead to a large adoption of the new product.

Kempe *et al.* [KKT03] first formalize the influence maximization problem. In this seminal paper, it defines two models, independent cascade (**IC**) model and linear threshold (**LT**) model, to simulate the influence spread. In addition, the authors prove that the problem is NP-Hard under both diffusion models, and develop

a general greedy framework which returns a seed set with $1 - 1/e - \epsilon$ approximation ratio, where ϵ is the error generated by using Monte Carlo simulation to estimate the influence spread. The general greedy algorithm is known to be inefficient in practice. However, considering the importance of the problem and the limitation in efficiency of the general greedy algorithm, it motivates a lot of follow-up works [LKG+07, CWY09, CWW10, CYZ10, GLL11b, GLL11a] which try to improve the performance under both models. However, these algorithms either sacrifice the effectiveness, *i.e.,* influence spread, or cannot scale well to very large graphs and large k. Also, there are some papers try to define different diffusion models, like the continuous time independent cascade model [GBS11]. In this chapter, we focus on the IC model and LT model, which are the most widely studied models in the literature. Under both models, the social network $\mathcal{G}$ can be considered as a distribution of a set of graph instances.

Recently, Borgs *et al.* [BBCL14a] develop an elegant framework, reverse influence sampling (**RIS**) to solve influence maximization problem. The intuition of the RIS framework is that given a randomly selected node v in $\mathcal{G}$ and a randomly selected instance g from $\mathcal{G}$ under a certain propagation model, let $R_g(v)$ denote the the set of nodes that can reach v in g. Then a set of nodes is more likely to have larger influence if it has higher probability to overlap with $R_g(v)$. Each $R_g(v)$ is called a *random reverse reachable set*, *i.e.,* a sample. The procedure of generating a sample can be divided into two phases: 1) sample a node v, 2) materialize the sample in a sampled instance g to get $R_g(v)$. The cost of the second phase is usually much larger than the first phase. To solve the influence maximization problem under the RIS framework, we first select a set $\mathcal{R}$ of samples and then iteratively select the node with largest marginal overlapping size with $\mathcal{R}$. This greedy strategy can return a result with $1 - 1/e - \epsilon$ approximation ratio, where ϵ is decided by the

number of samples selected.

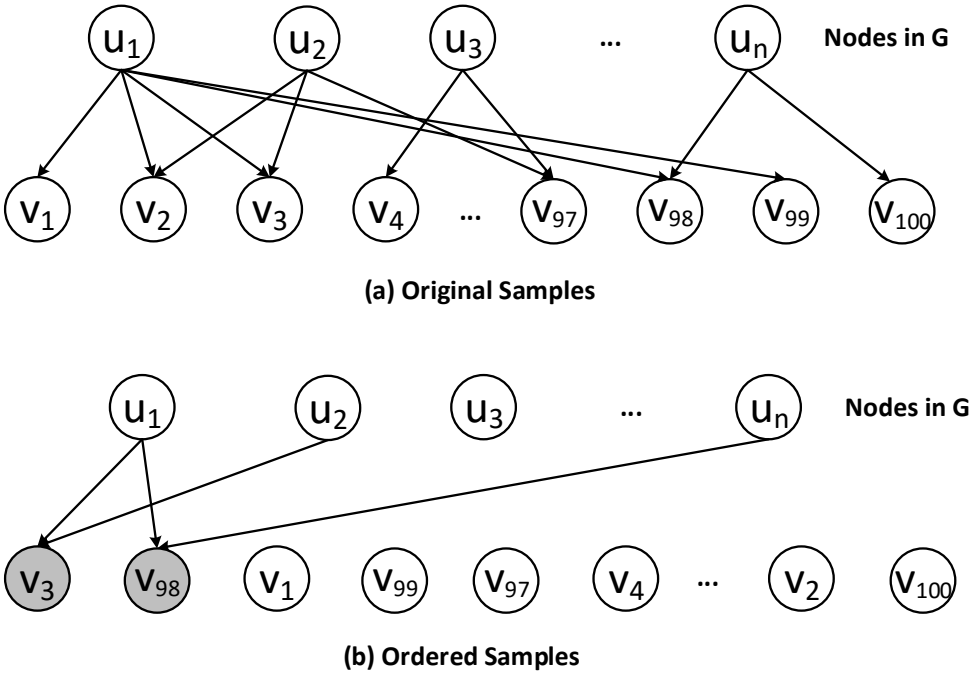


Figure 5.1: Motivation Example of Accelerating Nodes Selection

Consequently, the sample size directly determines the efficiency and effectiveness of the framework. In [BBCL14a], Borgs *et al.* conduct an in-depth theoretical analysis of the sample size needed to bound the effectiveness. However, due to the large constant factor in the sample size, it does not work well in practice. Tang *et al.* [TXS14] improve the sample complexity and show that the sample size needed is at least λ/OPT, where OPT is the influence spread of the optimal seed set and λ is an equation related to k and the error parameters. In addition, the authors provide several approaches to get a tight lower bound of OPT, which makes the RIS framework work well in practice and outperforms the previous studies. Tang *et al.* in [TSX15] propose the IMM method which is the state-of-the-art approach for influence maximization problem in both IC model and LT model. It utilizes the martingale technique to further reduce the sample size needed and reuse the

samples. IMM shows that the sample size needed is λ^*/OPT, where λ^* is smaller than λ. In addition, it provides an iterative approach to obtain a lower bound of OPT which is asymptotically tight without sacrificing the performance guarantee.

Although IMM offers strong performance in solving the influence maximization problem, it still leave much room for improvement in term of efficiency and scalability. In IMM, to determine a tight sample size, it needs to get a tight lower bound of OPT, which is a heavy overhead of the algorithm. This is because it needs to iteratively double the sample size and selects k nodes in the current samples to refine the lower bound of OPT. The cost cannot be neglected when k and n are large. In hence, the three approaches [BBCL14a, TXS14, TSX15] based on RIS can be implemented in a two-phase framework: 1) determine a sample size, 2) select k nodes based on the samples. Approach in [BBCL14a] is inefficient due to the large sample size, while TIM and IMM are limited by the heavy overhead in the first phase. In this chapter, instead of trying to obtain a tight sample size, we aim to accelerate the second phase over a reasonable large sample size, which is much more efficient to obtain, to speedup the processing. Following is a motivating example about how to speedup the second phase.

Example 5.1. *Given a set of samples, it corresponds to a bipartite graph, where the edges denote the reachability relationship between nodes u in $\mathcal{G}$ to the sampled nodes v_j. Suppose we sample 100 samples. Figure 5.1(a) is the corresponding bipartite graph based on the samples. Then u_1 is the first nodes selected since it covers the most number of samples. u_3 is the second node selected since it has the largest marginal coverage.*

Assume we have an oracle that assigns a total order to the samples. The order can provide the property that if a node covers more samples ranked in the front, it is more likely to have larger influence. Formally, the oracle provides a coverage

requirement r. It guarantees that if we go through the samples based on the order, the node first covers r samples will be the node with the largest coverage with high probability. As shown in Figure 5.1(b), we sort the samples in Figure 5.1(a) based on the oracle and r equals 2. Then we materialize the samples one by one based on the order. After materializing 2 samples v_3 and v_{98}, we find that u_1 is the first node covering 2 samples. Thus we can select u_1 as the first node. For the following samples, we remove them if they are covered by the nodes already selected. For example, we discard the samples v_1 and v_{99} when materializing them since they can be reached by u_1. For the following nodes selection, it equals the case of selecting the first one. For example, u_3 is the next one selected, since it covers v_{97} and v_4.

Based on the oracle, we can achieve possible early termination in nodes selection without materializing all the samples. Therefore, even if we cannot get a tight sample size needed, we can still run fast.

Contributions. Based on the motivating example, we bring the order of samples into the RIS framework to improve the performance. We propose a bottom-k sketch based RIS framework (BKRIS), which utilizes the bottom-k sketch to serve as the oracle in Example 6.3. In the worst-case scenario, if there are not enough (*i.e.,* less than k) nodes that can meet the oracle's requirement r, we still needs to materialize all the samples. So to guarantee the efficiency and the result quality, we develop a quick sample size estimation approach based on the small world property. Specially, we provide a cost-effective method that efficiently obtains a lower bound of OPT. By feeding our lower bound into the sample size equation in IMM (*i.e.,* λ^*/OPT), we can obtain a sufficient sample size needed. Also, we develop several optimizations to accelerate the generation of sample order and the processing of the worst-case scenario. We conduct extensive experiments on 10 real world social networks on the IC model and the LT model. For both models, BKRIS

achieves constantly speedup, up to two orders of magnitude, compared with IMM. In summary, our contributions are as follows.

- We propose the BKRIS framework, which accelerates the RIS framework by involving the order of samples based on bottom-k sketch.
- We propose an efficient method to derive a sufficient and reasonable large sample size by using the small world property.
- We provide novel techniques to optimize the generation of sample order and to efficiently handle the worst case.
- We experimentally evaluate BKRIS on 10 datasets, and show that we can achieve up to 2 orders of magnitude speedup compared with the state-of-the-art approach IMM on both the IC model and the LT model.

Road Map. The rest of the chapter is organized as follows. We briefly introduce the problem to be studied and the related techniques used in Section 5.2. In Section 5.3, we introduce the BKRIS framework, and describe the techniques developed. We demonstrate the efficiency and effectiveness of proposed framework in Section 5.4 on 10 real social networks. Lastly, we conclude the chapter in Section 5.5.

5.2 Background

In this section, we first formally introduce the influence maximization problem studied as well as the diffusion models utilized. Then we introduce the related techniques employed in this chapter. Table 5.1 summarizes the notations frequently used throughout the chapter.

Notation	Meaning
$\mathcal{G}$	a social network
$V(E)$	the set of nodes (edges) of $\mathcal{G}$
$n(m)$	the number of nodes (edges) in $\mathcal{G}$
u, v	a node or user in V
$\langle u, v\rangle$	a directed edge from u to v
S	a selected seed set $S \subseteq V$
$\sigma(S)$	the influence spread of set S
R	a random reverse reachable (RR) set
$\mathcal{R}$	a set of RR set
bk	parameter k in bottom-k sketch

Table 5.1: Summary of Notations

5.2.1 Problem Definition

We consider a social network as a directed graph $\mathcal{G} = (V, E)$, where V represents the set of nodes (users) and E denotes the set of edges (relationships between users) in $\mathcal{G}$. Let $|V| = n$ and $|E| = m$. Given an edge $\langle u, v\rangle \in E$, we say v is an outgoing neighbour of u and u is an incoming neighbour of v.

Diffusion Model. There are many models to simulate the influence propagation in a social network. In this chapter, we focus on the *independent cascade* (IC) model and the *linear threshold* (LT) model, which are most widely adopted by existing researches [CFL$^+$15, LZT15, BBCL14a, CDPW14b, CYZ10]. Note that the techniques proposed in this chapter are based on the RIS framework, thus they can be easily extended to support all the diffusion models that are supported by the RIS framework, such as the continuous time independent cascade model.

IC Model. In the IC model, each edge $\langle u, v\rangle$ has an independent probability $\mathcal{P}(u, v) \in [0, 1]$, indicating the probability that u can influence v. The influence diffusion process of a node set S works as following three steps.

- At timestamp 0, only the nodes in $S_0 = S$ are *active*, while all the other nodes are *inactive*.

- Let S_i denote the set of nodes that are activated at timestamp i. At timestamp $i+1$, each node $u \in S_i$ attempts to activate each of its inactive outgoing neighbour v with probability $\mathcal{P}(u, v)$.

- Once a node becomes active, it remains activated for subsequent iterations. The procedure terminates when there are no more nodes can be activated, *i.e.,* $S_t = \emptyset$, where $t = 0, 1, 2, \ldots$

The *influence spread* $\sigma(S)$ of S is defined as the expected number of nodes activated by the above procedure, *i.e.,* $\sigma(S) = \mathbb{E}[\sum_{i=0}^{t} S_i]$.

LT Model. Different from the IC model, the LT model is to simulate the case that a node is more likely to be influenced if most of its in neighbours are influenced. In the LT model, each edge $\langle u, v\rangle$ is associated with a weight $w(u, v)$ and $\sum_{u \in N_{in}(v)} w(u, v) \leq 1$, where $N_{in}(v)$ is the set of in neighbours of v. Given a set S of nodes, the first and the third steps of the LT model are the same as that in the IC model. In the LT model, the second step works as follows.

- Let S_i denote the set of nodes that are activated at timestamp i. At timestamp $i + 1$, a node $v \in V \setminus \cup_{0 \leq j \leq i} S_j$ is activated if $\sum_{u \in \cup_{0 \leq j \leq i} S_j} w(u, v) \geq \lambda_v$.

Similarly, the influence spread of S is the expected number of nodes activated in the whole procedure.

Problem Statement. Given a social network $\mathcal{G}$, a constant integer k and a probabilistic diffusion model $\mathcal{M}$, the problem of influence maximization is to find a set S^* of k nodes in G which has the largest influence spread, *i.e.,*

$$S^* = \arg\max_{S \subseteq V} \{\sigma(S) || S| = k\}$$

S^* is called a seed set and each node $u \in S^*$ is a seed. In this chapter, we study the case where $\mathcal{M}$ is set to IC model or LT model.

Problem Hardness. As shown in [KKT03], the influence maximization problem is NP-Hard in the IC and LT models, and Chen *et al.* [CWW10, CYZ10] have proved it is #P-Hard to calculate the influence spread of a seed set in both models. Fortunately, the influence spread function $\sigma(S)$ satisfies the following two properties.

- Monotonic property. For $S,\ T \subseteq V$ and $S \subseteq T$, we have $\sigma(T) \geq \sigma(S)$.
- Submodular property. For $S,\ T \subseteq V$, $S \subseteq T$ and $u \in V \wedge u \notin S \cup T$, we have $\sigma(S \cup \{u\}) - \sigma(S) \geq \sigma(T \cup \{u\}) - \sigma(T)$.

Based on these two properties, we can iteratively select the nodes with maximum marginal influence until k nodes have been found. It will return a result with an approximation ratio of $1 - \frac{1}{e}$, if the influence spread is exactly calculated.

5.2.2 Preliminaries

In this section, we first introduce the bottom-k sketch, and then we revisit the RIS based approaches.

Bottom-k Sketch

In this section, we briefly introduce the bottom-k sketch [CK07], which is used in our BKRIS framework to obtain the statistics information for early stopping. Bottom-k sketch is designed for estimating the number of distinct values in a multiset. Suppose N distinct points are uniformly distributed over $(0, 1)$, then the expected distance between any two adjacent points is $\frac{1}{N+1} \approx \frac{1}{N}$. Given a multiset $A = \{v_1, v_2, \cdots, v_n\}$ and a truly random hash function h, each distinct value v_i in the set A is hashed to $(0, 1)$ and $h(v_i) \neq h(v_j)$ for $i \neq j$. The bottom-k sketch consists of the k smallest hash values, *i.e.,* $\mathcal{L}_A = \{h(v_i) \mid h(v_i) \leq \mathcal{L}_A^k \wedge v_i \in A\}$,

where $\mathcal{L}_A^k$ is the k-th smallest hash value of the set. If $|A| \leq k$, we directly store all the elements in A and we can exactly calculate the number of distinct value. So if $A > k$ the number of distinct value can be estimated with Equation (5.1).

$$\hat{D} = (k-1)/\mathcal{L}_A^k \tag{5.1}$$

$\hat{D}$ is an unbiased estimator. Based on the analysis in [BHR+07], the expected relative error of the estimator is $\sqrt{2/\pi(k-2)}$ and the coefficient of variation of the estimation is no more than $1/\sqrt{k-2}$, which means the estimation converges fast with the increase of k. To distinguish from the value k used in influence maximization problem, hereafter in this chapter, we use bk to denote the parameter k in the bottom-k sketch.

The RIS based Approaches

Before introducing the RIS framework, we fist clarify some concepts for the ease of explanation.

Graph Distribution. The social network $\mathcal{G}$ under a probabilistic diffusion model $\mathcal{M}$ can be treated as a graph distribution $\mathcal{G} = \{g_i\}$. Each instance g_i in the distribution corresponds to a deterministic graph generated following the diffusion model $\mathcal{M}$. In the IC model, g_i is generated by flipping a coin on each edge $\langle u, v\rangle$, which is survived with probability $\mathcal{P}(u, v)$. In the LT model, for each node $v \in g_i$, we keep one living edge of its incoming neighbour or $\emptyset$ with probability $1 - \sum_{u \in N_{in}(v)} w(u, v)$.

Definition 5.1 (Reverse Reachable Set)**.** *Given an instance g_i from $\mathcal{G}$ and a node v, the reverse reachable (RR) set is the set of nodes that can reach v in g_i.*

Definition 5.2 (Random Reverse Reachable Set)**.** *An RR set is a random reverse reachable set if g_i is randomly sampled from $\mathcal{G}$ and v is randomly selected from all*

the nodes V.

In the chapter hereafter, we use RR set to denote random RR set for short if there is no ambiguity. The intuition of the RIS framework is that given a set $\mathcal{R}$ of RR sets, if a set of nodes appears in $\mathcal{R}$ frequently, it is more likely to have large influence spread. In addition, it is shown in [BBCL14b, TXS14] that Equation (5.2) is an unbiased estimation of $\sigma(S)$ for both IC and LT models, where $F(S, \mathcal{R})$ is the number of RR sets in $\mathcal{R}$ covered by S.

$$I(S) = n \times \frac{F(S, \mathcal{R})}{|\mathcal{R}|} \tag{5.2}$$

Therefore, we can generate a large number of RR sets and find the seed set that covers the maximum number of RR sets. Based on this intuition, the RIS framework can be summarized in Algorithm 6.

Algorithm 6: RIS Framework

Input : $\mathcal{G}$: a social network; k : number of seeds

Output: S : the seed set with size k

$\mathcal{R} \leftarrow \emptyset$; $S \leftarrow \emptyset$;

// Phase 1

Estimate a sufficient sample size θ ;

Generate RR set and insert into $\mathcal{R}$ **until** $|\mathcal{R}| = \theta$;

// Phase 2

while $|S| < k$ **do**

    Identify node v that covers the most number of RR sets in $\mathcal{R}$;

    $S \leftarrow S \cup \{v\}$;

    Remove all the RR sets from $\mathcal{R}$ covered by v ;

return S

The RIS framework consists of two phases. In the first phase, it tries to obtain a sample size θ that will be large enough to guarantee the approximation ratio for good estimation and returns a set $\mathcal{R}$ with θ RR sets. In the second phase it goes through k iterations, each time it adds the node with the maximum marginal coverage to the seed set.

In the three representative works [BBCL14a, TXS14, TSX15], the authors are all trying to fix the first phase, *i.e.,* reduce θ, while the second phase is identical. Borgs *et al.* [BBCL14a] use the number of edges visited when generating RR sets as the measure of sample sufficiency, but it has a large constant factor in the equation and does not work in practice. TIM/TIM+ [TXS14] concludes that θ should be at least λ/OPT to bound the performance. It also develops several approaches to obtain a tight lower bound of OPT to get a small sample size. IMM [TSX15] is the state-of-art method, which uses martingale to further reduce the size of λ and reuse the samples. So it can incrementally generates RR sets and estimate a tighter lower bound of OPT until the stop condition is satisfied. Then it adds enough samples to $\mathcal{R}$ instead of sampling θ RR sets from the beginning. IMM significantly outperforms TIM/TIM+ in practice.

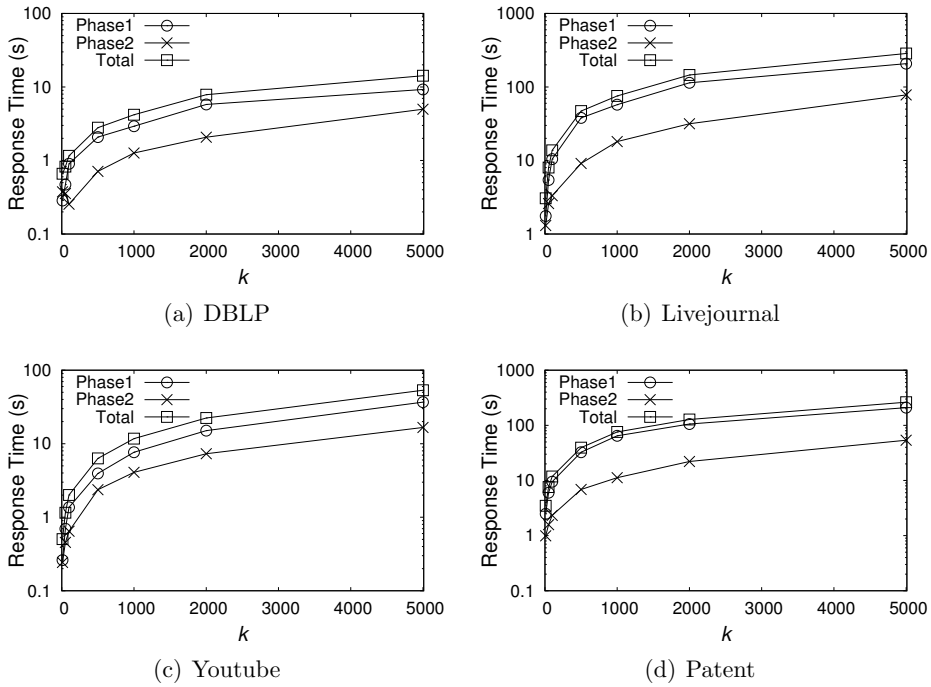
5.3 Bottom-k Based RIS Framework

In this section, we first introduce the motivation and the general framework of the BKRIS method. Then we describe the details of each component in the framework.

5.3.1 Motivation and General Framework

As stated in Section 5.2.2, the three representative works are focusing on developing novel techniques to reduce the sample size. However, even though IMM can reuse

the samples and significantly reduced the sample size compared with TIM/TIM+, the overhead of IMM is still large when k increases. As shown in Figure 5.2, we report the processing time of phase 1, phase 2 and the total time for the IMM method on four datasets by varying the seed set size k on the IC model. The experiment setting can be referred to Section 5.4 for more details. We can see the phase 1 of IMM becomes the bottleneck of its algorithm when k is large, and limits its scalability.



(a) DBLP (b) Livejournal

(c) Youtube (d) Patent

Figure 5.2: Overhead Evaluation for IMM

Motivated by the results, we may wonder if the overhead issue can be compromised, if we can quickly determine a reasonable large sample size and use the oracle in the motivating example. Thus it leads to our BKRIS framework. The BKRIS framework can also be divided into two phases as stated in Algorithm 7.

Quick Sample Size Estimation. Different from that of IMM, the aim of our

Algorithm 7: BKRIS Framework

Input : $\mathcal{G}$: a social network; k : a positive integer.

Output: S : the seed set with size k

1 $\theta \leftarrow$ QuickSampleSizeEstimation($\mathcal{G}$, k) ;

2 $S \leftarrow$ IncrementalSeedsSelection($\mathcal{G}$, θ, k) ;

first phase is to quickly estimate a sufficient sample size θ. The sufficiency of the sample size is defined as follows. If we conduct the RIS nodes selection algorithm with θ samples, it should return a seed set with $1 - 1/e - \epsilon$ approximation ratio with high probability. To fulfil the target, we develop an efficient algorithm which returns an exact lower bound of OPT and feed into the equation in IMM, *i.e.,* λ^*/OPT.

Incremental Seed Selection. In the seed selection phase, we incrementally materlize samples in a priority order and build the bipartite graph. We add a new node to seed set when its sketch information fulfils the requirement. We also offer several optimizations to accelerate the order generation process and reduce the cost for the worst-case scenario.

5.3.2 Quick Sample Size Estimation

In this section, we present an efficient approach to obtain a sufficient sample size θ for the IC model and the LT model.

Sample Size Requirement. According to the motivating example in Section 5.1, given the oracle, we can find the seed set without materializing all the samples. It seems that the sample size is not a big issue. We can simply set OPT to k in the sample size equation λ^*/OPT used by IMM. However, if there are only $k' < k$ nodes meet the oracle's requirement after going through all the samples, we still

needs to use the greedy algorithm to find $k-k'$ nodes under the RIS framework. In this case, we still need to materialize all the samples, thus the sample size θ should fulfil the following three requirements.

- The sample size cannot be too large. Otherwise, it will run slow in the worst case, like the approach in [BBCL14a].
- The sample size should be sufficient. The sufficiency here means the sample size should not be smaller than $\theta = \frac{\lambda^*}{OPT} = \frac{2n\cdot((1-1/e)\cdot\alpha+\beta)^2}{OPT\cdot\epsilon^2}$ [TSX15] (α, β are functions of l, k, n). Otherwise, we cannot claim that the algorithm will return a close influential seed set compared with the IMM method.
- The sample size should be computed efficiently, otherwise it will become the bottleneck of the whole algorithm. So we cannot use the method in IMM and TIM, in which retrieving a good sample size takes the most part of the cost.

Intuition. To meet the above requirements, we should efficiently obtain a lower bound of OPT and feed into the equation used by IMM. Note that, we just use the sample size equation in IMM, while our contribution here is providing a cost-effective method to obtain a lower bound of OPT.

In this chapter, we utilize the small world heuristics, that is we try to obtain the lower bound by exploring limited steps of a set of k nodes. This property is widely used by many heuristic approaches to solve influence maximization problem. In particular, we only use 2-hop neighbours of k nodes to guarantee the efficiency of the algorithm. When the diameter (*i.e.,* the longest path of the graph) of the social network is small, it is still time consuming to calculate the exact influence of k nodes in 2-hop neighbours under the IC model and the LT model. Thus we explore the k nodes in BFS manner, *i.e.,* we visit the nodes layer by layer and each

node will visit its out going neighbours once. This strategy is fast but it will lead to a slight drop compared with the exact influence to the 2-hop neighbours. So it is a lower bound of the influence of the k nodes, and it is clearly a lower bound OPT.

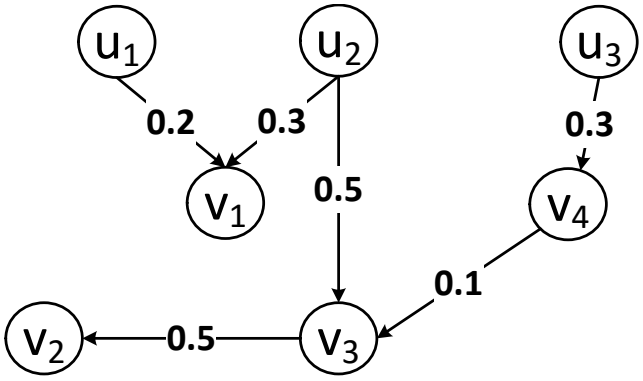


Figure 5.3: Example of Sample Size

Obtain Lower Bound of OPT in the IC Model. The details of the method on the IC model is presented in Algorithm 8. The input of the algorithm includes a social network $\mathcal{G}$ and a selected node set S with k nodes. Initially, we set the activated probability $prob[u]$ of all the nodes equal 0, except the nodes in S, whose probability is set to 1 in Line 1 and Line 2. H_0 and H_1 are two queues. H_0 stores the nodes in the zero layer, *i.e.,* the node set S. H_1 stores all the nodes that visited in the first layer of S. c is a counter that stores the total influence calculated with initial value as k. In Line 3, we mark all the nodes in H_0 as visited and the other nodes as unvisited. The mark is used to avoid pushing the same node into H_1 multiple times. Then we start to visit the nodes in H_0 one by one and push the node visited in the first layer into H_1 in Line 12. For each popped node u, we visit its each neighbour $v \in N(u)$. The cumulated influence on v is calculated as $(1 - prob[v])\mathcal{P}(u, v)prob[u]$, where $(1 - prob[v])$ is the probability that v is not activated by other nodes, and $\mathcal{P}(u, v)prob[u]$ is the probability that u activates v

based on u's current probability $prob[u]$ and influence from u to v. So the cumulated influence is the influence gain that brings from u to v. After processing the nodes in H_0, we continue the process on the nodes in H_1 until H_1 is empty. Based on the definition of the IC model, the algorithm returns a lower bound of $\sigma(S)$, so it is also a lower bound of OPT.

Algorithm 8: LowerBound

Input : $\mathcal{G}$: a social network; S : a set of k nodes.

Output: A lower bound of $\sigma(S)$ (IC Model)

```
1  for each u ∈ V do prob[u] ← 0 ;
2  for each u ∈ S do prob[u] ← 1 ;
3  mark nodes in H_0 as visited and the other nodes as unvisited ;
4  H_0 ← S; H_1 ← ∅; c ← k ;
5  while H_0 ≠ ∅ or H_1 ≠ ∅ do
6      if H_0 ≠ ∅ then
7          u ← H_0.pop() ;
8      else
9          u ← H_1.pop() ;
10     for each v ∈ N(u) do
11         if v is not visited and u is popped from H_0 then
12             push v into H_1 ;
13             mark v as visited ;
14         c+ = (1 − prob[v])P(u, v)prob[u] ;
15         prob[v]+ = (1 − prob[v])P(u, v)prob[u] ;
16 return c
```

Example 5.2. *Figure 5.3 is a small graph with probability (or weight for the LT model) on each edge. With k equal 3,* u_1, u_2 *and* u_3 *are the selected nodes. Initially* $H_0 = \{u_1, u_2, u_3\}$ *and* $prob[u_i] = 1$ *for* $i = 1, 2, 3$*. After exploring the first layer,* v_1*,* v_3 *and* v_4 *are pushed into* H_1 *and marked as as visited.* $prob[v_1] = 0.2 + (1 - 0.2) \times 0.3 = 0.44$*,* $prob[v_3] = 0.5$ *and* $prob[v_4] = 0.3$*. When exploring the second layer, we will encounter* v_2 *and* v_3*. If we encounter* v_2 *first,* $prob[v_2] = (1 - 0) \times 0.5 \times 0.5 = 0.25$*. If we encounter* v_3 *first,* $prob[v_3] = 0.5 + (1 - 0.5) \times 0.3 \times 0.1 = 0.515$ *and then the activating probability of* v_2 *will be* $prob[v2] = (1-0) \times 0.5 \times 0.515 = 0.2575$*. Thus, different visiting order will lead to slight drop on returned lower bound. However, it guarantees the algorithm will run fast, which is the major concern in our framework*

Obtain Lower Bound of OPT in the LT Model. For the LT model, the Algorithm 8 stills hold, except that we need to change the probability of the edge into the weight of the edge, and change Line 14 and Line 15 with the procedure in Algorithm 9. It is easy to verify the correctness of the algorithm based on the definition of LT model. Same as the problem in IC model, the algorithm may not return the exact influence spread in 2-hop neighbours. Hence it returns a lower bound of $\sigma(S)$ and OPT.

Algorithm 9: LowerBound(LT Model)

1 **if** $prob[v] = 1$ **then** continue ;

2 $c+ = \mathcal{P}(u, v)prob[u]$;

3 $prob[v]+ = \mathcal{P}(u, v)prob[u]$;

Example 5.3. *As shown in Figure 5.3,* v_1*,* v_3*,* v_4 *are on the first layer.* $prob[v_1] = 0.5$*,* $prob[v_3] = 0.5$*,* $prob[v_4] = 0.3$ *after visiting the first layer. If we first visit* v_2 *in the second layer, then* $prob[v_2] = 0.25$*,* $prob[v_3] = 0.53$ *after visiting the second layer.*

Choose the Nodes. To select a set of k nodes to feed into Algorithm 8, we use the degree based heuristic method. The reason we do not use the simple degree discount approach introduced in [CWY09] is that when the degree of nodes is large, degree discount will incur more update cost and need to do k times selection operation to obtain the final k nodes. It will be more cost when k is large. Therefore, we use the degree heuristic to select the k nodes with the largest degree. Specially, we use the unordered partial sorting approach [Hoa61], as we don't care about the order of these k nodes.

Discussion. The algorithm for obtain a lower bound of OPT runs in $O(m+n)$ time and the quick selection approach runs in $O(n)$ time on average, so the time complexity of sample size estimation is $O(m+n)$.

5.3.3 Incremental Seed Selection

In this section, we introduce the incremental seed selection method by using the bottom-k sketch. Given sample size θ, we aim to find k seeds that cover the most samples. In addition, we introduce several optimizations to accelerate the searching process and reduce the space. We start by introducing the procedure for selecting the first seed, then extend it to select k seeds.

Select the First Seed

Given the sample size θ, we randomly select θ nodes $\{(i,v)|i=1,2,\cdots,\theta\}$ from $\mathcal{G}$, where i is used as the id for this sample in case we sample the node v multiple times. Note that, we do not materialize the sample here. So for each sample the sample cost is $O(1)$ currently. For each node in $\mathcal{G}$, we maintain a counter initialized with 0. For each sample (i,v), we randomly generate a hash value $h(i,v) \in (0,1)$, and sort the samples in increasing order of their hash value. We process the sample

one by one in order. Let (i, v) be the current sample. We first retrieve the RR set $R(i, v)$. For each node $u \in R(i, v)$, we increase the counter of u by 1. When a node's counter reaches bk, which is the statistics pre-defined, we return it as the first seed. If there is no node's counter reaching bk after processing all the samples, we return the one with the largest counter as the first seed. According to Lemma 5.1, the above procedure is correct based on bottom-k sketch estimation.

Lemma 5.1. *The procedure of selecting the first seed is correct based on bottom-*k *sketch estimation.*

Proof. Suppose u_1 is the node selected by the procedure after processing sample (i, v_1), which means the sketch size of u_1 reaches bk first and the bk-th hash value equal $h(i, v_1)$. $bk-1/h(i, v_1)$ is an unbiased estimation of u_1 coverage on the sample. Continue processing the following samples, if another node u_2's counter reaches bk after processing sample (j, v_2). We have $h(j, v_2) > h(i, v_1)$ since we process samples according to the order of their hash value. So the estimation size of u_1 is larger than u_2, and the lemma is correct □

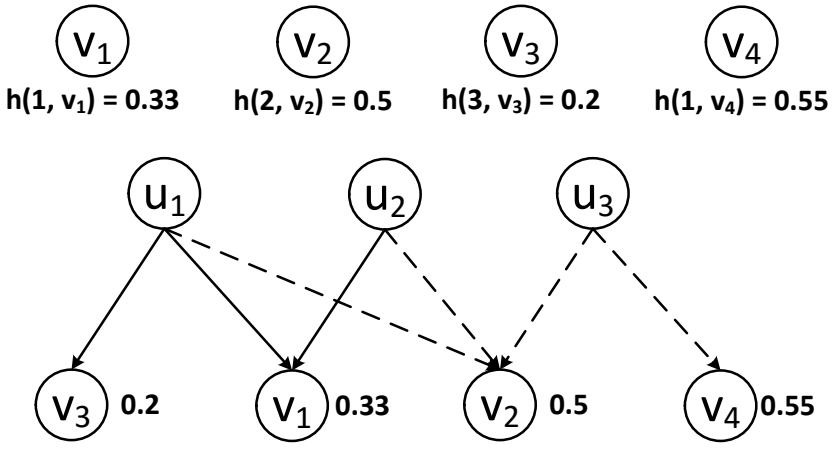


Figure 5.4: Example of Seed Selection

Example 5.4. *As shown in Figure 5.4, the sample size θ equals 4 and nodes $v_1 - v_4$ are sampled. We generate four hash values for them as listed. bk is set as 2. The bipartite graph shown in figure is the case if we have materialized all the samples' RR sets, and the samples are sorted by their hash values. We process the sample one by one in the order of their hash values. The samples linked with solid line denote that we materialize their RR sets, while the samples with dotted line are not materialized. As we go through the sample in order, u_1 is the first sample with counter reaching bk, so it is selected as the first seed with estimation size $(2-1)/0.33 = 3$. If we continue checking samples, u_2 will be the second node with counter reaching bk. u_2's bk-th hash value is 0.5 which is larger than that of u_1 0.33. So u_1 should be the first seed to be selected.*

The Coefficient of Variation of the estimation no more than $1/\sqrt{k-2}$. So the estimation converges fast with bk, and we can obtain the node with the largest coverage with high probability if bk is chosen properly. Thus if k equals 1, we can terminate the algorithm when a node's counter reaches bk. In this case, we save the cost of constructing the bipartite graph and the cost of doing linear scan to find the node with the largest coverage.

Optimization of Sample Order Generation. Based on the selection procedure, in our problem, we only aim to select the first seed instead of estimating the influence of a node, so we only care about the order of the samples based on their hash values. It is easy to verify that if the hash values of samples are removed and only sample order is kept, the procedure of selecting the first seed is still correct. Hence, we can replace the Strategy 1 in Figure 5.5 with a random shuffle of the selected samples, *i.e.,* Strategy 2 in the figure. The FisherYates shuffle algorithm [FY38] needs to generate θ random values and runs in $O(\theta)$ time, which reduces the cost of sorting samples.

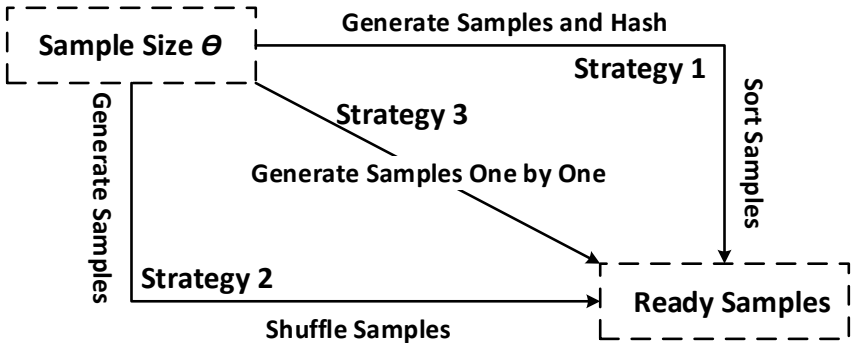


Figure 5.5: Sample Order Generation

However, the sample size θ can be very large in real applications, especially when k increases. Moreover, in sample size estimation phase, we do not aim to obtain a tight lower bound of OPT. According to Strategy 2 in Figure 5.5, we still need to generate $2 * \theta$ random variables: θ for getting samples and the other θ for shuffling the samples. Even it runs in linear time, it is still memory consuming, in particular, we may even not go through all the samples eventually. According to Lemma 5.2, the Strategy 2 is equal to Strategy 3 in Figure 5.5, *i.e.*,, we only need to sample a node when needed instead of sampling θ nodes in advance.

Lemma 5.2. *The data distribution returned by the sample-shuffle strategy equals the data distribution returned by the directly sampling strategy.*

Proof. To prove the distribution of this two procedures are equal, it is equivalent to proving the probability of any node falling on the same position in the ordered sample is the same. Put it another way, the probability of generating the same ordered samples is identical. Given a node $v \in V$ and a position p_i of the ordered samples where $1 \leq p_i \leq \theta$. Let $\mathcal{P}_2(v, p_i)$ and $\mathcal{P}_3(v, p_i)$ be the probability of a sample node v falling on the position p_i based on Strategy 2 and Strategy 3, respectively.

Strategy 2. Since it is sample with replacement, v can appear in θ multiple times.

The probability that v appears in θ i times is shown in Equation (5.3).

$$\mathcal{P}^i(v) = \binom{\theta}{i} (\frac{1}{n})^i (1 - \frac{1}{n})^{\theta - i} \tag{5.3}$$

Suppose the node v appears in the samples i times, then the probability of v appearing in p_i equals to $\frac{i}{\theta}$. Since for a random shuffle, the probability of each sample appears in any position is equivalent. Thus we have:

$$\begin{aligned}\mathcal{P}_2(v, p_i) &= \sum_{j=1}^{\theta} \mathcal{P}^j(v) \times \frac{j}{\theta} \\ &= \sum_{j=1}^{\theta} \binom{\theta}{j} (\frac{1}{n})^j (1 - \frac{1}{n})^{\theta - j} \times \frac{j}{\theta} = \frac{1}{n}\end{aligned}$$

Procedure 3. Since each sample is independent with others, any node appears in p_i with equal probability. We have $\mathcal{P}_3(v, p_i) = \frac{1}{n}$.

$\mathcal{P}_2(v, p_i) = \mathcal{P}_3(v, p_i)$, so the lemma is correct. □

Based on Lemma 5.2, we only need to generate samples when needed, instead of using $O(\theta)$ space in advance. In this case, we save both the space and time for generating the order of sample.

Select Subsequence Seeds

Based on the idea of selecting the first seed, we can easily extend it for selecting k seeds. The general idea is that for each node u we maintain a list records the samples u can reach so far and a counter for the list. After identifying the first node u', for the other nodes, we remove the samples covered by u' and reduce their counters accordingly. For each following sample (i, v), if its RR set contains the nodes in the found seeds, we just discard it. In this case, the remaining condition is equivalent to selecting the first seed. Algorithm 10 describes the details of selecting k seeds.

The input of the Algorithm 10 consists of a sample size θ, a social network $\mathcal{G}$ and the required seed set size k. The algorithm can be divided into two procedures. The first one is bottom-k based seed selection and the second is a clear-up procedure to deal with the case when we cannot select k seeds by using the first procedure.

In Line 1 and Line 2, we initialize the variables for the algorithm. q is a counter that counts the number of RR sets sampled. $\mathcal{L}(u)$ records the samples that u reaches so far and $C(u)$ is its corresponding counter. When q is smaller than θ and $|S|$ is smaller than k, we continue retrieving samples. Line 4 and Line 5 can be combined into one process, *i.e.,* when we try to obtain the RR set of a sample, if we meet a node in S, we discard this sample and go to the next iteration. This is a safe discard, since the sample is covered by S. Otherwise, we update the counter and the list of u in the RR set in Line 6. If a node's counter reaches bk, we select it as the next seed and remove all the nodes covered by it from other nodes' lists in Line 8–13. When we cannot select k seeds after the first procedure, it goes to the clean-up phase, which is simply to select the nodes one by one based on the nodes counters in Line 16. After selecting a node, it also needs to update the list and counter for the other nodes in Line 17–20.

Example 5.5. *Following Example 5.4, we assume $k = 2$ in this case. After processing the second samples, the counter $C(u_1)$ of u_1 reaches bk and selected as the first seed. After updating, the counter of u_2 is equal to 0. Then we process the third sample v_2. Since u_1 is in its RR set, we stop retrieving its RR set when we meet u_1, and continue to the next iteration. After processing the last sample, the counter of u_3 is $1 < bk$, so we go to the clean-up procedure. Finally, u_3 is selected as the second seed, since it has the largest counter among the rest nodes.*

Optimization of Clean-up Procedure. If we cannot select k seeds in the first phase, we have to go to the clean up procedure. Since we need to do $k - |S|$

iterations to select the seeds with maximum counter among n nodes, which is time consuming when n and $k - |S|$ are large. As the counters of nodes are updated in each iteration, a simple sort is not working. However, the reason we go into the clean-up procedure is that none of the node's counter reaches bk, which means the counters of the rest nodes are all smaller than bk. Thus we can only maintain bk lists from 0 to $bk - 1$, each list stores the nodes with their counter equal to the list number. Then we only need to go through the list from number $bk - 1$ to 0 to find the rest seeds. The optimized clean up method is described in Algorithm 11.

By scanning the n nodes once, we partition the nodes into bk lists in Line 2 according to their counters. Then we go through these lists from with large counter to small. If the current list is not empty, we obtain a node from the list and add it to the seed set from Line 5 to Line 8. Then we update the counters of nodes and move them to the new list with list number equal their counters in Line 14. The process terminates when $|S|$ is equal to k. The movement can be fast if implemented with linked list. It can run faster if we maintain bk arrays of size n when enough memory is available.

Example 5.6. *We replace the clean-up procedure in Example 5.5 with the optimized one. After going to the optimized clean-up, we first scan the nodes, and create two lists:* $L_1 = \{u_3\}$ *and* $L_0 = \{u_2\}$. *Then we obtain the next seed from list* L_1.

Discussion. The expected time complexity of the seed selection phase is equal to $O(\theta|R|)$ in the worst case, when we need to materialize all the samples to find the seed set, where $|R|$ is the expected size of an RR set. According to Algorithm 10, in each iteration it is identical the case to find the nodes with the largest coverage (*i.e.,* largest marginal coverage), so the returned seed set can guarantee a small error in sample coverage compared with the results from directly doing RIS algorithm on the same sample size, when bk is well chosen.

5.3.4 Summary

Since in the sample size estimation phase, we develop an exact lower bound of OPT without any approximation and feed into the equation in [TSX15] to get the sample size θ. If we conduct the RIS method directly on θ samples, it should return a seed set with $1 - 1/e - \epsilon$ approximation ratio with $1 - n^{-l}$ probability, according to the analysis in [TSX15]. In the second phase, based on sample size θ, it returns a set of k nodes with competitive performance on sample coverage and its success probability proportional to bk. So combine the two phase together, the BKRIS algorithm can return a seed set of k nodes with $1 - 1/e - \epsilon - \epsilon_1$ approximation ratio with probability $1 - n^{-l}$, where $\epsilon_1 = O(1/\sqrt{bk})$ is the error involved by the bottom-k sketch. As shown in the experiment, the influence spread of the returned seed set converges fast with bk, and we set $bk = 16$ by default in the experiments.

5.4 Experiments

In this section, we present the results of a comprehensive performance study on the IC model and the LT model to evaluate the efficiency and effectiveness of the proposed techniques in this chapter.

5.4.1 Experiment Setup

Algorithms. We compare our algorithm with the **IMM** algorithm [TSX15] which is the state-of-the-art method for both the IC model and the LT model. The parameters ϵ and l on the IMM method are set to 0.5 and 1, respectively, which is the default setting in [TSX15]. ϵ and l is set to the same value in our algorithm. The bk statistics is set as 16 by default, and we also investigate the impact of bk in the empirical study.

Datasets. To demonstrate the effectiveness and efficiency of our method, we conduct experiments on 10 real world social networks. The details of the datasets are reported in Table 6.2, and we provide a short name for each dataset. The Nethept dataset is obtained from [TSX15], and all the other 9 datasets are downloaded from SNAP [1]. The largest dataset used in the experiments is Friendster (FR), which has more than **1.8 billion** edges.

Diffusion Model and Work Load. We demonstrate the performance of the proposed algorithms on the IC model and the LT model then compare with the IMM method. For the IC model, the probability on each edge $\langle u, v\rangle$ is set as $\frac{1}{indeg(v)}$, where $indge(v)$ denotes the number of incoming edges of the node v. For the LT model, the weight of each edge $\langle u, v\rangle$ is set as $\frac{1}{indeg(v)}$. These are widely used setting in literatures [CWY09, CYZ10, TSX15]. We vary the seed set size k form 10 to 5000, with 5000 as default to demonstrate the scalability of the evaluated methods. To verify the algorithms' effectiveness, we conduct 10,000 round Monte Carlo simulations on each returned seed set to calculate its influence spread. On each setting and dataset, we run each algorithm five times and take the average as the final results.

Implementation and Environment. The source code of the IMM method is obtained from the authors. For our algorithm, we use the same basic data structure as IMM for the fairness of the comparison. All experiments are carried out on a PC with Intel Xeon 2.30GHz and 96G RAM. The operating system is Redhat. All algorithms are implemented in C++ and compiled with GCC 4.8.2 with -O3 flag.

[1] `https://snap.stanford.edu/data/`

5.4.2 Parameter Tuning

In this section, we present the results on the tuning of the parameter bk in our algorithm. Experiments are conducted on two datasets, Epinions and Gowalla, for the IC model and the LT model. As analyzed previously, the influence spread of BKRIS should converge rapidly with the increase of bk. We vary bk from 4 to 64 in the experiments.

We first present the results on the IC model. The response time and the influence spread are reported in Figure 5.6. Note that BKRIS-X represents the BKRIS algorithm whose bk is set to X. In Figure 5.6(c) and 5.6(d), as the increase of bk, the response time grows. From Figure 5.6(a) and 5.6(b), we can find that the effectiveness of BKRIS converges fast with bk. When bk reaches 8, the drop of the performance becomes less significant. In Figure 5.7, similar observation is reported from the experiments for the LT model. Thus, in the following experiments, we set bk to 16.

5.4.3 Results on the IC Model

In this section, we present the experiment results conducted on the IC model.

Results on all the Datasets. We first conduct the experiments on all the 10 datasets. For the ease of comparison on different datasets, we report the **effectiveness ratio** and **speedup ratio** of two algorithms. In particular, the effectiveness ratio and speedup ratio of IMM are defined in Equation 5.4, and that of BKRIS are always set to 1.

$$\begin{aligned} \text{Effectiveness ratio} &= \frac{\text{influence spread of IMM}}{\text{influence spread of BKRIS}} \\ \text{Speedup ratio} &= \frac{\text{response time of IMM}}{\text{response time of BKRIS}} \end{aligned} \tag{5.4}$$

In Figure 5.8, we evaluate the effectiveness ratio on 10 datasets with k to be

5,000. We also report the influence spread of BKRIS, which is marked on the top of bar of BKRIS. The effectiveness ratio of IMM is rather close to 1 for most of the datasets, and the only noticeable gap is from Patent dataset, on which the influence spread of BKRIS and IMM is 270,492 and 278,900, respectively. This shows that two methods have the similar performance in terms of influence spread. Nevertheless, Figures 5.9 reports that BKRIS outperforms IMM on running time by a wide margin, up to two orders of magnitude speedup, on all datasets. The running time (in second) of BKRIS on each dataset is marked on the top of the corresponding bar. These results suggest that BKRIS is superior to IMM because BKRIS can achieve the similar influence spread with much less running time. Take the largest dataset FR as an example, IMM takes 3664.97 seconds to identify the seed set, while BKRIS only uses 36.66 seconds to achieve the same influence spread.

Results on Selected Datasets. We also report more results on six selected datasets with different scales. In particular, Nethept and Gowalla represent the small graphs, Youtube and Patent represent the medium graphs, and Orkut and Friendster represent the large graphs. The influence spread and response time are reported in Figure 5.10 and 5.11 by varying k from 50 to 5000. As shown in Figure 5.10, two algorithms achieve the similar influence spread except there is a slight drop on the Patent dataset. However, BKRIS significantly outperorms IMM under all the settings, achieving the speedup of up to two orders of magnitude.

5.4.4 Results on the LT Model

In this section, we report the experiment results conducted on the LT model.

Results on all the Datasets. In Figure 5.12 and 5.13, we report the effectiveness ratio and speedup ratio on the 10 datasets under the LT model with k to be 5000. We place the influence spread and response time of BKRIS on the top of

its corresponding bar. Similar to the results reported for the IC model, on the LT model the effectiveness ratio is almost 1 for all the datasets, except a slight drop on the Patent dataset. However, BKRIS achieves more than 2 orders of magnitude speedup compared with IMM. It is observed that, under the same setting, the algorithms have longer response time compared with their performance on the IC model. This is because in order to generate a random RR set for a chosen node, we need to flip a coin for each relevant edge in the IC model while this is imposed to each relevant node in the LT model.

Results on Selected Datasets. In this set of experiments, we report the performance of two algorithms on the selected datasets used in the IC model by varying k from 10 to 5000. The results are reported in Figure 5.14 and 5.15. Same as the IC model, BKRIS runs much faster than IMM regarding all k values and meanwhile can achieve almost the same influence spread. It is noticed that in Figure 5.14(d), the difference of influence spread on the Patent dataset becomes smaller compared with the results on the IC model, but BKRIS still maintains significant speedup over IMM. In Figure 5.15(f), it only takes 0.55 seconds for BKRIS to return 50 seeds, while IMM uses 28.93 seconds.

5.4.5 Summary

Our comprehensive experiments clearly demonstrate the superiority of BKRIS compared to the state-of-the-art technique IMM on two popular influence models. Particularly, among the 10 real network datasets, BKRIS consistently reach the similar influence with IMM, while runs up to two orders of magnitude faster. It is also observed that BKRIS is more scalable towards the k value and the size of the network, which makes BKRIS more applicable to handling big network data emerging in the era Big Data.

5.5 Conclusion

In this chapter, we investigate the influence maximization problem. As we study, the state-of-the-art method [TSX15] still has limitations in scalability in both k and dataset size. In this chapter, we provide a more efficient solution, BKRIS, which integrates the bottom-k sketch with the RIS framework. Particularly, we bring the order of the samples into the RIS framework, which enables us to achieve possible early termination before materializing all the samples. In addition, we can incrementally identify seed set, which avoids the cost of linear scan k times to find the seeds. To bound the quality of returned seeds, we propose an efficient heuristic approach to obtain a lower bound of OPT. We also propose several optimizations to deal with sample order generation and the worst case processing. We conduct extensive experiments on 10 real social network datasets. Compared with IMM, we can achieve up to 2 orders of magnitude speedup.

Algorithm 10: IncrementalSeedSelection

Input : θ : RR set sample size; $\mathcal{G}$: a social network; k : a positive integer.

Output: a seed set of size k

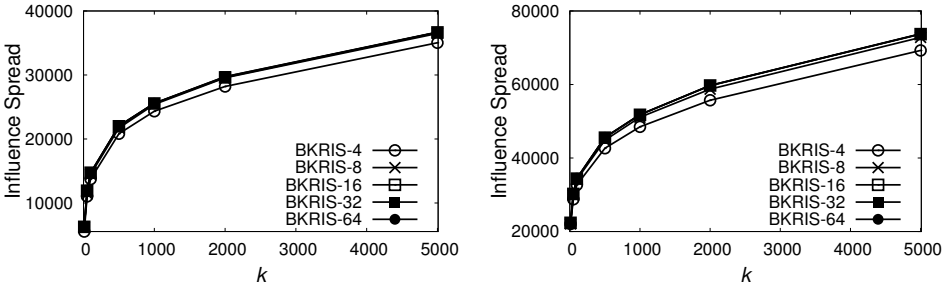
```
1  S ← ∅; q ← 0 ;
2  for each u ∈ V do L(u) ← ∅; C(u) ← 0 ;
   // Bottom-k based Seed Selection
3  while q ≤ θ and |S| < k do
4      Sample a random RR set R_i ;        /* retrieve R_i is stopped when
       meeting a node in S */;
5      if R_i ∩ S ≠ ∅ then continue ;
6      for each u ∈ R_i do L(u) ← L(u) ∪ {i}; C(u) + + ;
7      for each u ∈ R_i do
8          if C(u) = bk then
9              for each j ∈ L(u) do
10                 if R_j is not visited then
11                     Mark R_j as visited ;
12                     for each u' ∈ R_j do C(u') − − ;
13             S ← S ∪ {u}; break ;
14     q + + ;
   // Clean-Up
15 while |S| < k do
16     S ← S ∪ u with the largest C(u) ;
17     for each j ∈ L(u) do
18         if R_j is not visited then
19             Mark R_j as visited ;
20             for each u' ∈ R_j do C(u') − − ;
21 return S
```

Algorithm 11: CleanUpOptimization

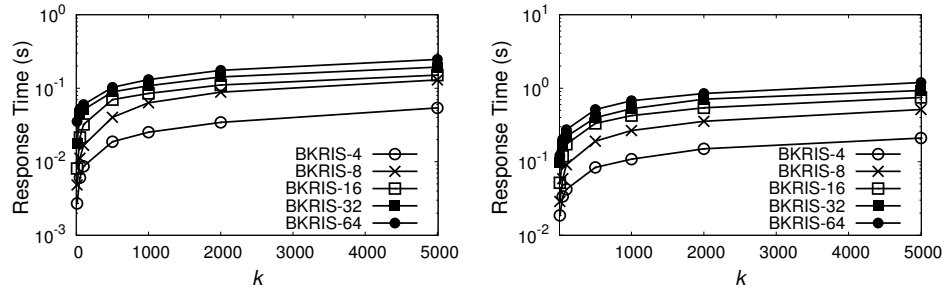
```
for i from 0 to bk − 1 do L_i ← ∅ ;
for u ∈ V \ S do L_{C(u)} ← L_{C(u)} ∪ {u} ;
p ← bk − 1 ;
while |S| < k do
    if L_p = ∅ then
        p − − ; continue ;
    u ← L_p.pop() ;
    S ← S ∪ {u} ;
    for each j ∈ L(u) do
        if R_j is not visited then
            Mark R_j as visited ;
            for each u′ ∈ R_j do
                C(u′) − − ; ;
                move u′ to L_{C(u′)} ;
```

Dataset	n	m
Nethept (NE)	15,229	62,752
Epinions (EP)	75,879	508,837
DBLP (DB)	317,080	2,099,732
Gowalla (GO)	196,591	3,801,308
Youtube (YT)	1,134,890	5,975,248
Patent (PA)	3,774,768	16,518,948
Pokec (PO)	1,632,803	30,622,564
Livejournal (LJ)	4,847,571	68,993,773
Orkut (OR)	3,072,441	234,370,166
Friendster (FR)	65,608,366	1,806,067,135

Table 5.2: Summary of Datasets

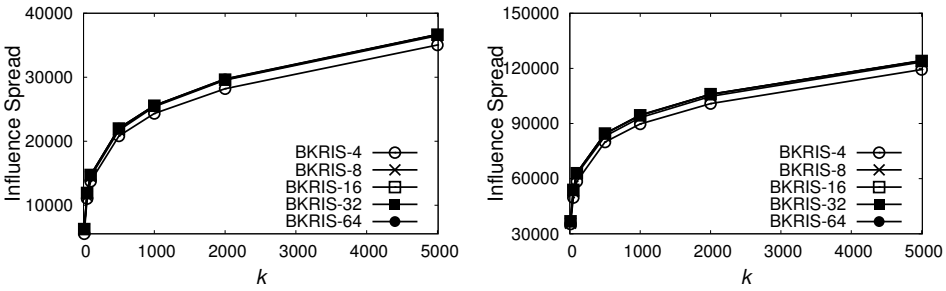


(a) Epinions (b) Gowalla

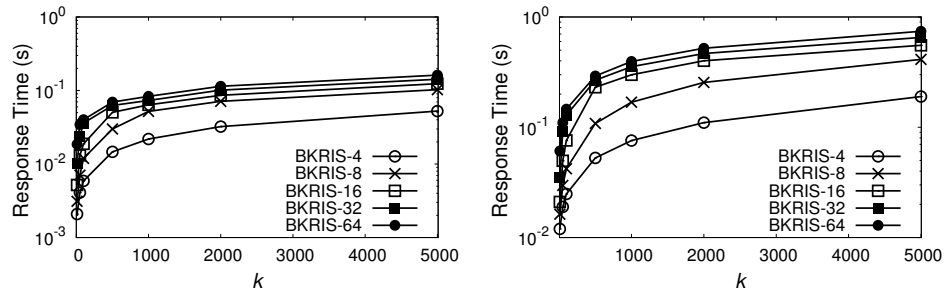


(c) Epinions (d) Gowalla

Figure 5.6: Tuning bk on the IC Model



(a) Epinions (b) Gowalla



(c) Epinions (d) Gowalla

Figure 5.7: Tuning bk on the LT Model

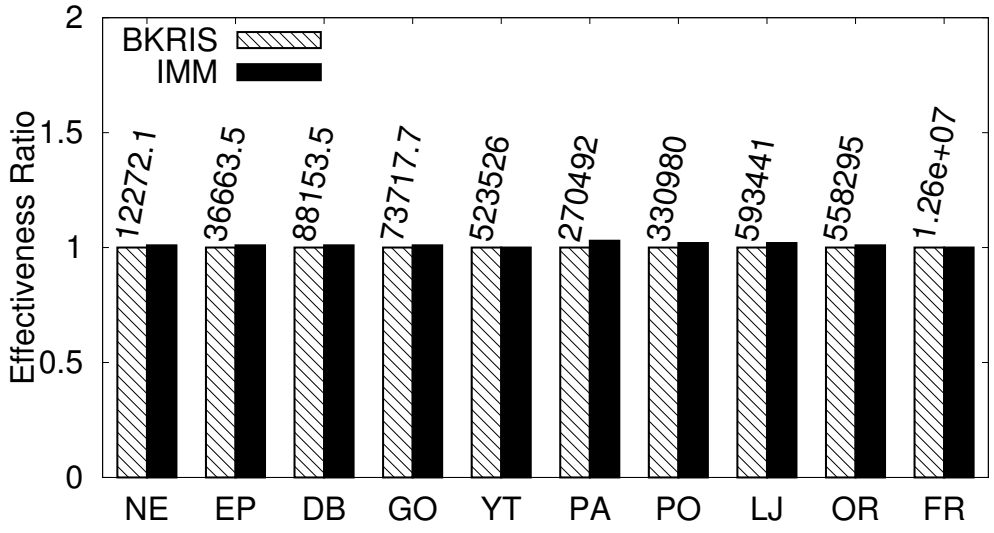


Figure 5.8: Effectiveness Ratio on the IC Model (the influence spread of BKRIS is marked on the bar)

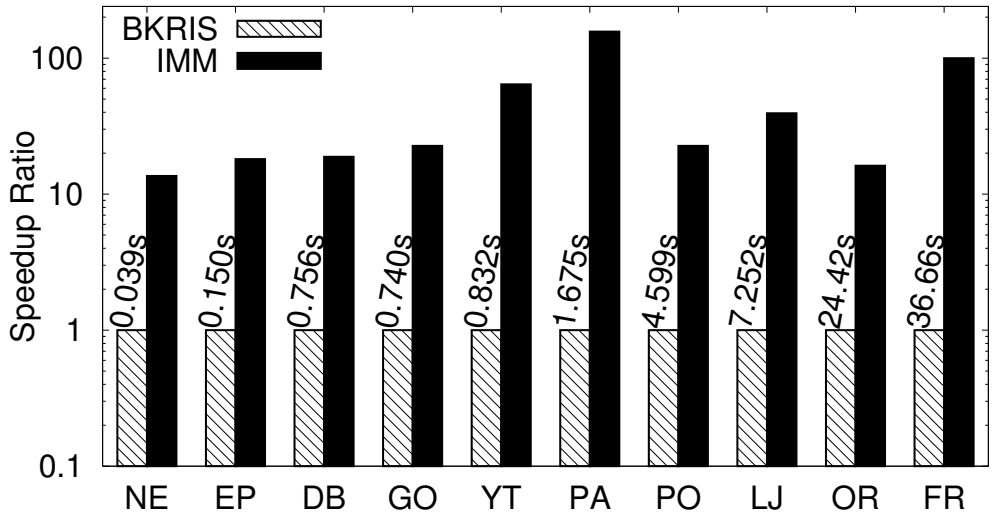
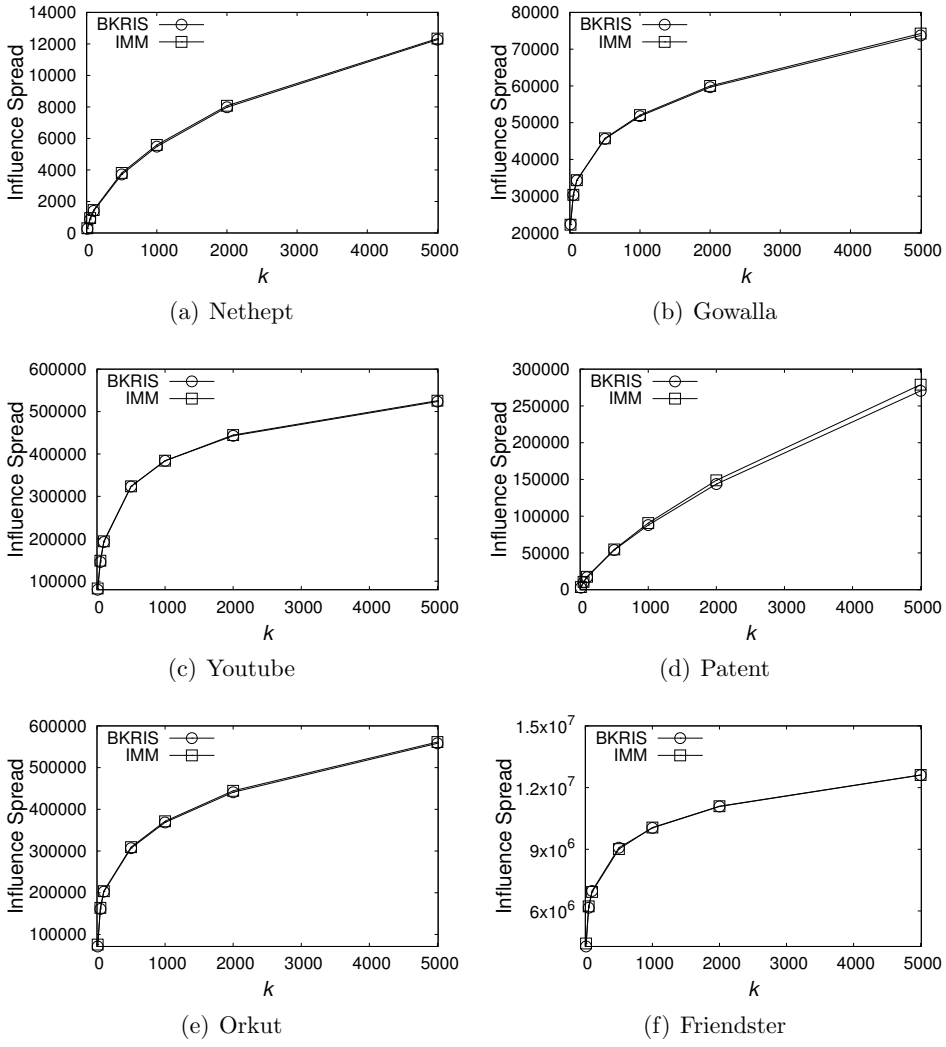


Figure 5.9: Speedup Ratio on the IC Model (the response time of BKRIS is marked on the bar)

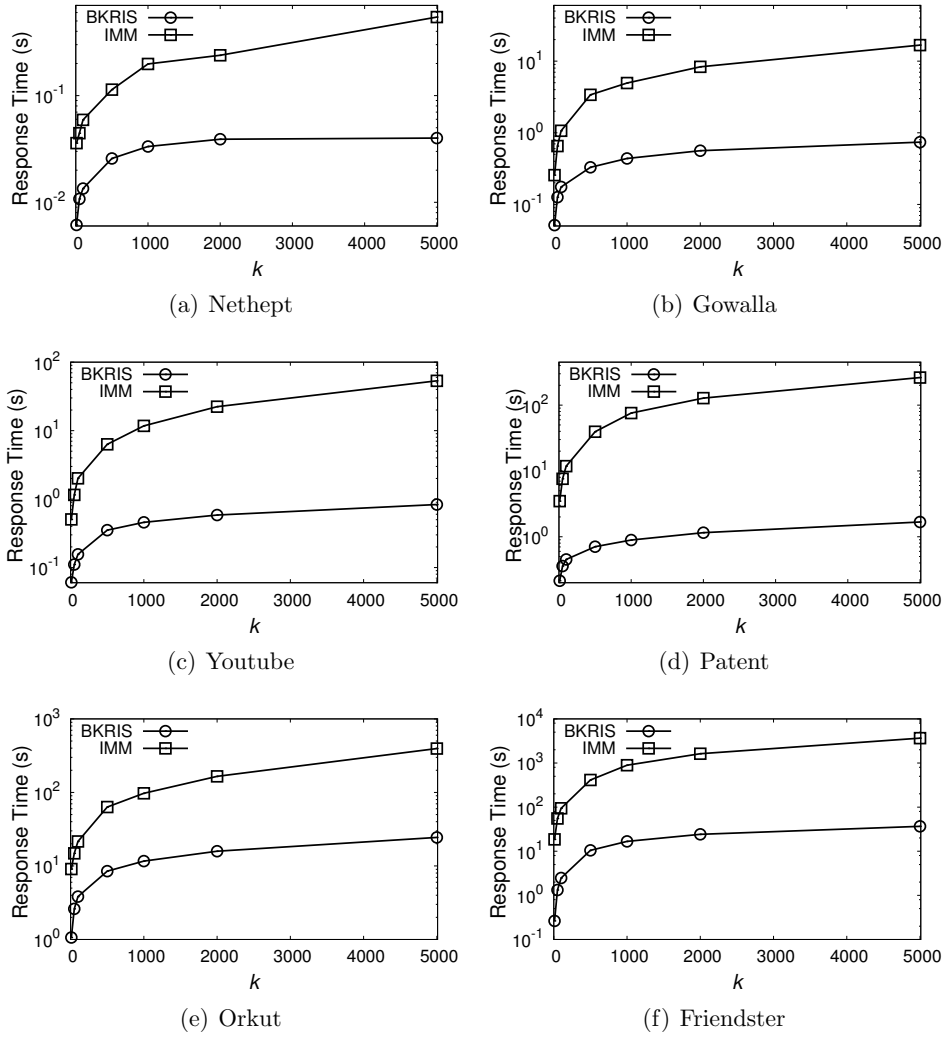


(a) Nethept (b) Gowalla

(c) Youtube (d) Patent

(e) Orkut (f) Friendster

Figure 5.10: Influence Spread on the IC Model



(a) Nethept (b) Gowalla

(c) Youtube (d) Patent

(e) Orkut (f) Friendster

Figure 5.11: Response Time on the IC Model

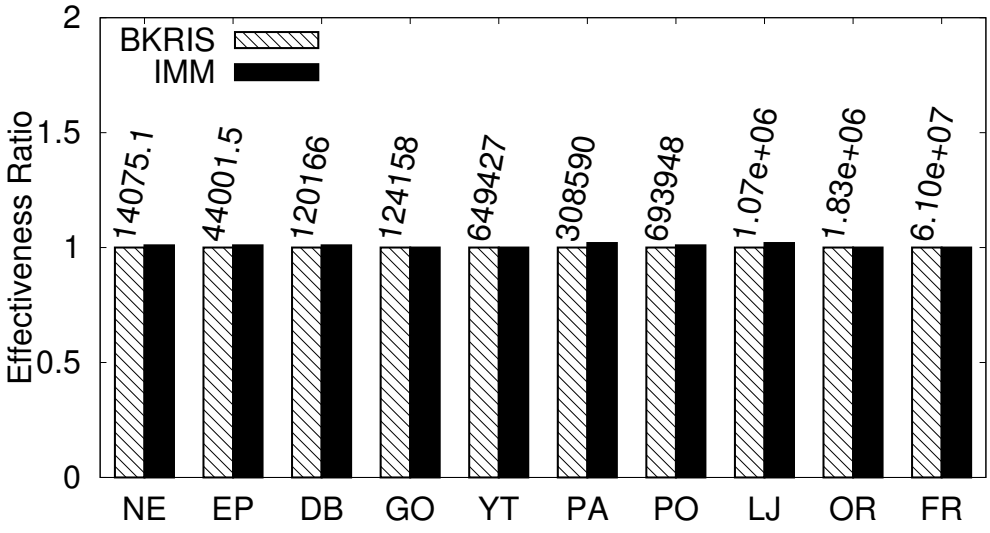


Figure 5.12: Effectiveness Ratio on the LT Model (the influence spread of BKRIS is marked on the bar)

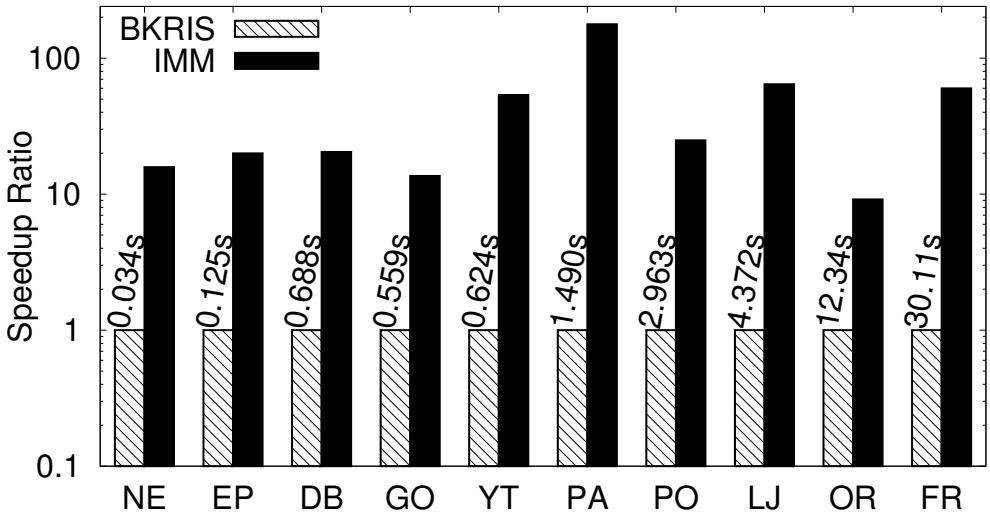
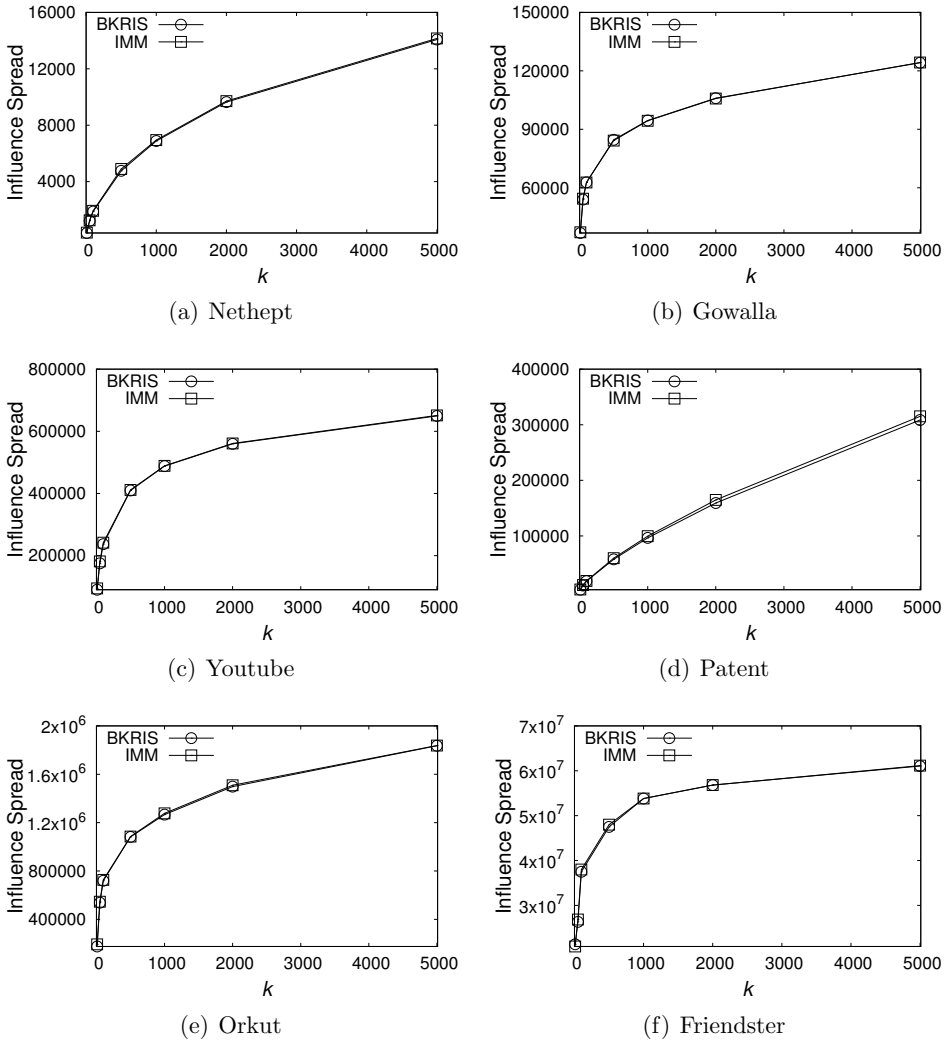


Figure 5.13: Speedup Ratio on the LT Model (the response time of BKRIS is marked on the bar)

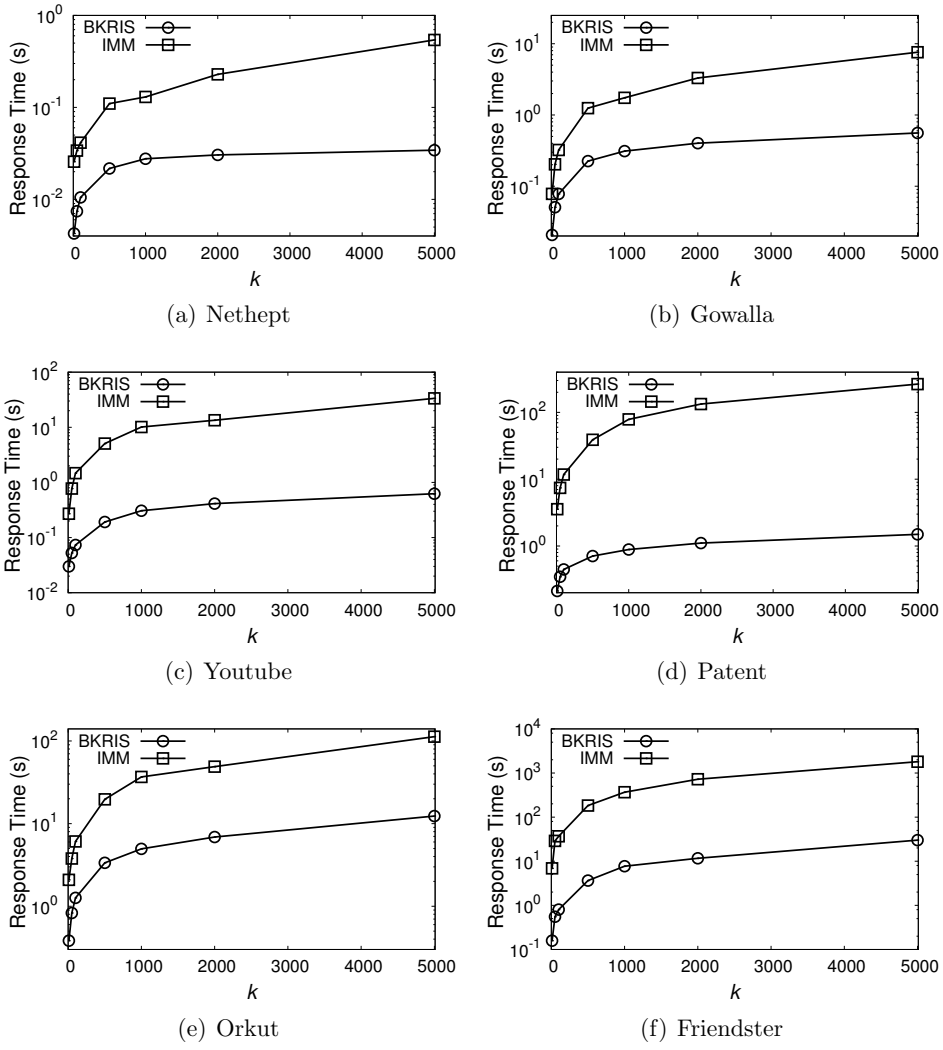


(a) Nethept (b) Gowalla

(c) Youtube (d) Patent

(e) Orkut (f) Friendster

Figure 5.14: Influence Spread on the LT Model



(a) Nethept (b) Gowalla

(c) Youtube (d) Patent

(e) Orkut (f) Friendster

Figure 5.15: Response Time on the LT Model

Chapter 6

Maximum Closeness Centrality Group Identification

6.1 Overview

As a subject of broad and current interest, social networks have been widely studied for decades. A social network is usually represented as a graph $G = (V, E)$ where V denotes the set of nodes (users) and E denotes the set of edges (relationships among users). Centrality, which measures the importance of a node in a social network, has been a fundamental concept investigated in the social networks. There are different measurements of centrality developed for various purposes, such as closeness centrality [Bav50], betweenness centrality [AGW15], eigenvector centrality [BL15a], etc. In this chapter, we focus on the classic closeness centrality, which is defined as the inverse of the average distance from a node to all the other nodes in the social network. The distance between two nodes is calculated by the shortest path distance. The smaller the average distance of a node is, the more important or more influential the node will be. To find the influential nodes (users) in a social

network, many research efforts have been made to find the k nodes with the largest closeness centrality [EW01, OCL08, OLH14]. However, in many real applications, such as team formation, we may need to find a set of k users which has large closeness centrality as a *group*, instead of returning the k independent users in the top-k ranking. In this chapter, we extend the definition of closeness centrality for a single node to a set of nodes. Specifically, the closeness centrality of a set S of nodes is defined as the inverse of the average distance from S to the nodes in G. And the distance from S to a node $u \in V$ is defined as the minimum distance from u to the nodes in S. The maximum closeness centrality group identification problem is to find a set of k nodes in the social network with the largest closeness centrality.

The challenges of the problem lie in two aspects. First, we show that the problem is NP-Hard, thus there is no polynomial time solution unless P=NP. Fortunately, we prove that the objective function is monotonic and submodular. It means we can obtain a result with $1-1/e$ approximation ratio by adopting a greedy framework. Second is that we still need the information of the all pairs shortest path distances even for the simple greedy algorithm, which is prohibitive to compute ($O(|V|^3)$ time) and store ($O(|V|^2)$ space) when the graph is large. There are some efficient index such as the network structure index techniques [RMJ07]. It partitions the graph into zones and approximates the shortest path distance between two nodes by using their distances to the same zone. However, this approximation offers no guarantee for the quality of the returned distance. In order to scale to large graphs, we propose a sampling based approach by extending the traditional sampling method for estimating the closeness centrality of a single node. In addition, we bring order into the samples, such that the nodes can be identified incrementally. Then we utilize the selected nodes to reduce the cost of computing the distances from the nodes to the samples. To further accelerate the process, we

develop optimization techniques to reduce the updating cost for the less important nodes. Through experiments on real world social networks, we verify the efficiency and effectiveness of the proposed techniques.

The rest of the chapter is organized as follows. Section 6.2 formally introduces the problem studied in this chapter as well as the greedy algorithm based framework. Section 6.3 presents the proposed sampling based algorithms. We demonstrate the efficiency and effectiveness of the proposed techniques on four real social networks in Section 6.4 and conclude the chapter in Section 6.5.

Notation	**Meaning**
G	a social network
$V(E)$	the set of nodes (edges) of $\mathcal{G}$
k	number of selected nodes
$n(m)$	the number of nodes (edges) in G
u, v	a node or user in V
S	a selected seed set with $S \subseteq V$
$c(S)$	closeness centrality of S
$\hat{c}(S)$	estimation of $c(S)$
$\mathcal{L}$	a set of samples with $\|\mathcal{L}\| = l$

Table 6.1: Summary of Notations

6.2 Background

We formally define the problem of maximum closeness centrality group in this section. Then we analyze the properties of the objective function and introduce our greedy algorithm based framework. Table 6.1 summarizes the notations frequently used throughout the chapter.

6.2.1 Problem Definition

We consider a social network $G = (V, E)$ as an undirected connected graph, where V denotes the set of nodes, and E denotes the set of edges. $|V| = n$ and $|E| = m$. For each edge, we assign a weight to it, denoting the distance between the two nodes. For nodes $u, v \in V$, the distance $d(u, v)$ between the two nodes is calculated as the shortest path distance and $d(u, u) = 0$. Then the classic closeness centrality of a node u is defined by the inverse average distance from u to the nodes in G.[1] By extending the classic closeness centrality for a single node, we can define the closeness centrality for a set of nodes as follows.

Definition 6.1 (Closeness Centrality for a Set of Nodes)**.** *Given a social network G and a set S ($S \subseteq V$) of nodes, the closeness centrality of S is denoted by $c(S)$, which is measured by the inverse average distance from S to all the nodes in G, i.e.,*

$$c(S) = \frac{n}{\sum_{v \in V} d(S, v)}$$

where $d(S, v) = \min\{d(u, v)\}$ for $u \in S$.

Problem Statement. Based on Definition 6.1, we define the *maximum closeness centrality group identification* (**MCGI**) problem as follows. Given a social network G and a positive integer k, the MCGI problem aims to find a set S^* of k nodes that has the largest closeness centrality, *i.e.,*

$$S^* = \arg\max_{S \subseteq V} \{c(S) \mid |S| = k\}. \tag{6.1}$$

The selected node set is call *seed set* and each node in the set is called a *seed*. According to Lemma 6.1, the MCGI problem is NP-Hard.

[1]Note that there are different variants of the definition of closeness centrality, in this chapter we focus on the classic closeness centrality.

Lemma 6.1. *The maximum closeness centrality group identification problem is NP-Hard.*

Proof Sketch. The correctness of Lemma 6.1 can be proved by reducing from a known NP-hard problem "k-means clustering problem in the Euclidean space" [ADHP09]. The polynomial time reduction can be summarized as follows. Each object in the k-means clustering corresponds to a node in the social network. The k centers in the k-means clustering problem correspond to the selected k nodes. The Euclidean distance between objects corresponds to the shortest path distance between nodes in MCGI. Even if we can compute the shortest path distance in constant time, the hardness of the problem still remains the same. □

6.2.2 Objective Function Analysis

According to Lemma 6.1, we know that there is no polynomial time solution for the MCGI problem. Fortunately, based on Lemma 6.2, the closeness centrality function for a set of nodes shown in Definition 6.1 has the following two properties.

- Monotonicity. Given any two sets $S,\ T \subseteq V$ with $S \subseteq T$, a function $f(x)$ is monotonic, if $f(S) \leq f(T)$.
- Submodularity. Given any two sets $S,\ T \subseteq V$ with $S \subseteq T$ and $v \in V \setminus T$, a function $f(x)$ is submodular, if $f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T)$.

Lemma 6.2. *The closeness centrality function for a set of nodes is monotonic and submodular.*

Proof. Assume there are two sets satisfying $S,\ T \subseteq V$ with $S \subseteq T$.

- **Monotonic**. Since the shortest path distance from a set S of nodes to a single node v is decided by the minimum distance from nodes in S to v, we have

$d(S,u) \geq d(T,u)$ for $u \in V$. Thus we have $\sum_{u \in V} d(S,u) \geq \sum_{u \in V} d(T,u)$, which means $c(S) \leq c(T)$. Consequently the monotonic property is correct.

- **Submodularity**. Given two nodes u, v, where $v \in V \setminus T$ and $u \in V$, we divide the proof into two conditions. (1) If $d(T,u) > d(T \cup \{v\}, u)$, it means the distance from u to v is smaller than that of u to any node in T, so is for S. Then we must have $d(T \cup \{v\}, u) = d(S \cup \{v\}, u) = d(v,u)$. In hence, we have $d(S \cup \{v\}, u) - d(S,u) \leq d(T \cup \{v\}, u) - d(T,u)$. (2) If $d(T,u) = d(T \cup \{v\}, u)$, it means u is closer to nodes in T. Thus we have $d(S \cup \{v\}, u) - d(S,u) \leq d(T \cup \{v\}, u) - d(T,u) = 0$. Then $\frac{1}{c(S \cup \{v\})} - \frac{1}{c(S)} \leq \frac{1}{c(T \cup \{v\})} - \frac{1}{c(T)}$ holds. Thus the submodular property holds based on the definition of closeness centrality.

□

6.2.3 Greedy Algorithm Framework

Since the objective function is monotonic and submodular, we can utilize the greedy algorithm to iteratively select the node with the largest marginal gain (*i.e.,* the node will increase the closeness centrality of the set most by adding it). According to the proof of Nemhauser *et al.* [NWF78], for submodular monotone functions, any s prefix of the greedy sequence has the centrality score that is at least $1-(1-1/s)^s \geq 1-1/e$ of the optimal score, which means the greedy algorithm provides a bounded approximation to the optimal solution of the NP-hard problem. Therefore, through k iterations it can return a set S of k nodes with $1-1/e$ approximation ratio, *i.e.,* $c(S) \geq (1-1/e)c(S^*)$, where S^* is the optimal result. The details of the greedy algorithm framework are shown in Algorithm 12.

In Algorithm 12, S maintains the set of nodes already selected in the previous iterations. M is a matrix that stores the distance $d(S \cup \{u\}, v)$ $(v \in V)$ for each

Algorithm 12: GreedyFramework

Input : G : a social network, k : a positive integer.

Output: S : a set of k nodes

```
1  S ← ∅ ;
2  M ← all pairs shortest path distances ;
3  Score ← {c(u) | u ∈ V};
4  while |S| < k do
5      v = arg max_{w∈V\S} Score[w] ;
6      S ← S ∪ {v} ;
7      for each u ∈ V \ S do
8          for each w ∈ V do
9              if d(u, w) > d(v, w) then
10                 M[u, w] = d(v, w) ;
11         Score[u] ← c(S ∪ {u}) ;
12 return S
```

node u. *Score* is a vector that maintains the closeness centrality of adding u to S, *i.e.,* $c(S \cup \{u\})$ for each node $u \in V$. And $Score[u]$ equals $c(u)$ at the beginning. After initialization, we iteratively select the node with the largest marginal gain from $V \setminus S$ in Line 5. After selecting the node v with the largest marginal gain, we add it to S and update the distance matrix M as well as the *Score* value for each node $u \in V \setminus S$ from Line 7 to 10. The procedure is repeated until we find k nodes.

Example 6.1. *As shown in Figure 6.1, the weight of each edge is marked on the edge. Initially, we compute the all pair shortest path distance:* $d(v_1, v_2) = 1, d(v_1, v_3) = 1, d(v_1, v_4) = 1$, $d(v_2, v_3) = d(v_2, v_1) + d(v_1, v_3) = 2$, $d(v_2, v_4) = d(v_2, v_1) + d(v_1, v_4) = 2$, $d(v_3, v_4) = d(v_3, v_1) + d(v_1, v_4) = 2$, *and the initial Score for*

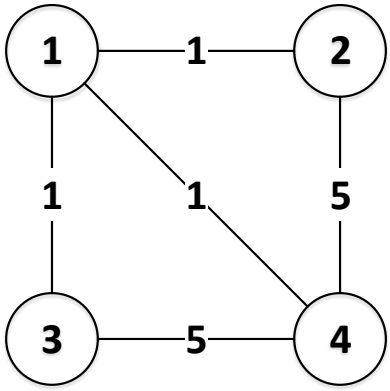


Figure 6.1: Greedy Algorithm

each node is calculated: $c(\{v_1\}) = 4/3, c(\{v_2\}) = 4/5, c(\{v_1\}) = 4/5, c(\{v_1\}) = 4/5$. *Thus* v_1 *is the first node selected.*

Analysis. The space complexity of Algorithm 12 is $O(n^2)$, since we need to store the all pairs distances in M. To compute the all pairs shortest paths, we can use the Floyd-Warshall algorithm which needs $O(n^3)$ time. Note that we can also use other algorithms to compute the all pairs shortest paths but it is not the major concern in this chapter. In the node selection phase, we need $O(kn^2)$ time to select the k nodes as we need to update the distance matrix and the marginal gain for all the unselected nodes after each iteration.

6.3 Sampling based Algorithms

Although the greedy approach in Algorithm 12 can return a result with a bounded approximation ratio, *i.e.,* $1 - 1/e$, it suffers from serious limitations when scaling to large graphs. As analyzed previously, it needs $O(n^2)$ space to store the all pairs distances, which is prohibitive for large graphs. If we do not store the all pairs distances in memory and only compute them on the fly when needed, the computation

time is still not affordable for large graphs. Moreover, we need considerable amount of time to update the marginal gain for each node in each iteration. Therefore, we propose two sampling based approaches to make it possible to scale to large graphs in this section.

6.3.1 Baseline Sampling Method

To make it possible for processing large graphs, a natural consideration is to utilize the sampling techniques. It is able to obtain a high quality result by accessing only a small part of the graph's information. In the previous works, sampling based approaches are developed to estimate the closeness centrality of a single node, or to identify the top-k closeness centrality nodes. The idea of estimating the closeness centrality of a single node u can be summarized as follows. We first randomly sample several nodes $\mathcal{L}$ without replacement from V. Then we calculate the shortest path distances from u to all the samples $v \in \mathcal{L}$. Next we use the average distance from u to all the samples as an estimation of the average distance of u to all the nodes. The estimator is presented in Equation (6.2).

$$\hat{c}(u) = \frac{l}{\sum_{v \in \mathcal{L}} d(u, v)} \tag{6.2}$$

The sample size $l = |\mathcal{L}|$ is usually much smaller than n in order to get a good estimation. Through this way, we can quickly estimate the closeness centrality of a node with theoretical guarantee [CDPW14a, CCK15, Coh14]. Motivated by the idea, we can use the average distance from a set S of nodes to all the samples as the estimation of the average distance of S to all the nodes. It can be formally expressed in Equation (6.3).

$$\hat{c}(S) = \frac{l}{\sum_{v \in \mathcal{L}} d(S, v)} \tag{6.3}$$

where $\mathcal{L}$ denotes the sample set of size l. It is easy to verify $1/\hat{c}(S)$ is an unbiased estimation of $1/c(S)$, *i.e.,* $1/c(S) = \mathbf{E}\left[1/\hat{c}(S)\right]$.

Based on the estimator, we can modify Algorithm 12 to apply the sampling technique and solve the MCGI problem. The idea is to use the estimated closeness centrality and marginal gain to replace the exact calculated value in the greedy framework. To be more specific, we first sample l nodes from V. For the distance matrix M, it stores the distances from each node $u \in V$ to all the samples, *i.e.,* $d(S \cup \{u\}, v)$ $(v \in \mathcal{L})$. Initially, it only stores the distances from all the nodes to the l samples. This can be done by running a single source shortest path algorithm from every sample. Similarly, $Score[u]$ stores $\hat{c}(S \cup \{u\})$. Then we iteratively select k nodes with the largest marginal gain based on the sampling method. The pseudo-code is omitted due to the space limitation.

Example 6.2. *As shown in Figure 6.1, suppose the sample size is v_4 is the sampled node. We first compute the distance from other nodes to v_4 and compute the estimated closeness centrality,* i.e., $\hat{c}(v_1) = 1, \hat{c}(v_2) = 1/2, \hat{c}(v_3) = 1/2$. *While the estimated influence for v_4 is $+\infty$, since the distance to itself is 0.*

Analysis. The space cost is $O(nl)$ for the baseline sampling algorithm, since only the distances from all the nodes to the l samples are maintained. In order to compute the initial matrix M, we can run l times single source shortest path algorithm (*e.g.,* Dijkstras algorithm) which requires $O(l(m + n \log n))$ time. We also need $O(kln)$ time to select the k nodes. Due to the sampling techniques, the result will have $1 - 1/e - \epsilon$ approximation ratio, where ϵ is the error introduced by sampling. From previous studies [Ind99, Tho01, EW01] we know the estimation procedure converges quickly with the sample size l. Thus both space and processing time are reduced compared with Algorithm 12.

6.3.2 Order based Sampling Method

Although the baseline sampling algorithm can greatly reduce the space cost and time complexity, there is still room for improvement.

- The first limitation is that it strictly separates the shortest path distance calculation phase and the node selection phase. As introduced later, we can further reduce the cost if incrementally doing the sampling and the seed selection.

- The second limitation is that after selecting a node, it needs to update the marginal gain for all the nodes, which is costly when n is large. Since most of the nodes are insignificant, *i.e.,* with small closeness centrality, we can save time if we can avoid updating their marginal gain.

Based on the observations, we present two optimized techniques to further reduce the cost of baseline sampling algorithm.

Incremental Sampling and Node Selection. Incremental sampling is aiming to tackle the first limitation. Before introducing the details techniques, we first explain the motivation. Suppose we plan to sample l samples in total, based on the first $l' < l$ samples nodes, we already select some nodes into S (using l' samples). Then we can use these selected nodes (seed set) to reduce the calculation cost of shortest path distances for the rest $l - l'$ samples. Following is a motivating example.

Example 6.3. *As shown in Figure 6.2, assume $k = 2$, $l = 10$ and the weight of each edge is 1. We first sample 5 samples and adopt the 5 samples into the baseline sampling framework, and we select the first seed v_4. In hence currently $S = \{v_4\}$. Next, we examine the rest 5 samples to find the second seed. Suppose v_3 is the next*

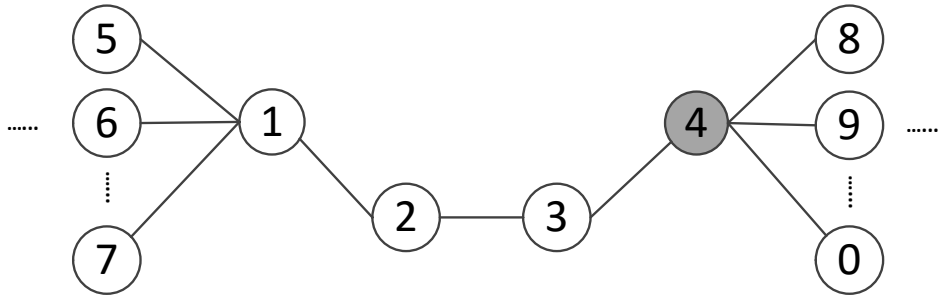


Figure 6.2: Motivating Example of Incremental Sampling and Node Selection

sampled node, then we start running a Dijkstra's algorithm from v_3. The property of Dijkstra's algorithm is that it will incrementally reach the nodes close to the source node. So in the figure, v_3 first reaches nodes v_2 and v_4 with shortest path distance of 1. Remember that v_4 is the node already selected. Then we do not need to further compute the distance from all the other nodes to v_3, as their distances are greater than that of v_4. The reason is that adding any of them to S will not change the distance from S to the sample v_3. Consequently, their distances to v_3 are all recorded as $1 = d(v_3, v_4)$, that is for $v' \in V \setminus \{v_2, v_3, v_4\}$, it holds $d(S \cup \{v'\}, v_3) = 1$. We can apply this strategy to other sampled nodes to reduce the computation cost.

According to the motivating example, if we can incrementally do the sampling and node selection, we will significantly reduce the cost of computing shortest path distances from nodes to the samples. To fulfill it, we firstly divide the sample into k partitions $\{P_1, P_2, ..., P_k\}$. For the i-th node selection, we compute the distances from all the nodes to the samples in P_i by utilizing the selected $i-1$ nodes. For a sample in P_i, we start a Dijkstar's algorithm from it and we can terminate the shortest path computation, if we reach a selected node. We repeat this procedure for all the samples in P_i and we can obtain the node with the largest marginal gain by considering the estimation due to samples in $\cup_{j=1}^{i} P_j$. The details of the algorithm are shown in Algorithm 13.

Algorithm 13: OrderBasedSampling

Input : G : a social network, k : a positive integer, l : sample size.

Output: S : a set of k nodes

1 $S \leftarrow \emptyset$; $Score \leftarrow \emptyset$; $M \leftarrow \emptyset$;

2 Get samples and partitions $\{P_1, P_2, ..., P_k\}$; `/* partition of` l `samples */`;

3 **for** i from 1 to k **do**

4 Compute the distance from all nodes to P_i with S as constraint ;

5 Update $Score$ for $u \in V \setminus S$ based on the samples in $\cup_{j=1}^{i} P_j$;

6 $v = \arg\max_{w \in V \setminus S} Score[w]$;

7 $S \leftarrow S \cup \{v\}$;

8 **for each** $u \in V \setminus S$ **do**

9 **for each** $w \in \cup_{j=1}^{i} P_j$ **do**

10 **if** $d(u, w) > d(v, w)$ **then**

11 $M[u, w] = d(v, w)$;

12 **return** S

In Algorithm 13, we sample l nodes and divide them into k partitions in Line 2. To get l samples without replacement, we can do a random permutation on all the nodes and select the first l nodes. In such case, each sample has a rank induced by the permutation. Then each partition can store the samples based on the order, *i.e.,* P_i stores the $(i-1)l/k$-th to il/k-th sampled nodes.

In the i-th iteration, we compute the distances from all the nodes to samples in P_i with S as the constraint. It means when running Dijkstra's algorithm from a sample v', we can early stop if we get the shortest path distance from v' to a seed node u' in S. In Line 5, by considering the distance computed from the sampled

nodes in P_i, we update the $Score$ value for nodes based on the samples in $\cup_{j=1}^{i} P_j$. Then we select the node v with the largest marginal gain and add it to S. Finally, we update the distance matrix, *i.e.,* $d(S \cup \{u\}, w)$ for $u \in V \setminus S$ and $w \in \cup_{j=1}^{i} P_j$. The algorithm terminates after k iterations.

Correctness of Algorithm 13. The aim of the algorithm is to select the node with the largest marginal gain from $V \setminus S$ in each iteration. The input of the iteration is the seed set S from previous $i-1$ iterations. Since the proposed optimization only prunes the unnecessary calculation, the distance matrix M and $Score$ vector are exactly calculated based on the samples in $\cup_{j=1}^{i} P_j$ given S. Moreover, $\cup_{j=1}^{i} P_j$ consists of the first $|\cup_{j=1}^{i} P_j|$ samples of the l samples, as we partition the samples based on the sample order (*i.e.,* permutation order). Consequently, the case in iteration i amounts to selecting the node with the largest marginal gain, based on the already selected $i-1$ nodes with $|\cup_{j=1}^{i} P_j|$ samples. Therefore the algorithm is guaranteed to be correct. Although the incremental selection strategy may affect the accuracy of estimation, the quality drop is negligible as shown in the experimental evalutions.

Update Optimization. In the baseline sampling algorithm, after selecting one node in an iteration, we need to update the $Score$ vector for all the other nodes. However, the closeness centrality tends to be very small, *i.e.,* insignificant nodes, for most of the nodes in a social network. Usually we have $k << n$. As a result, the nodes with small centralities will never be selected into the results. If we can avoid updating their marginal gains we will save a large amount of computational cost. Following is a motivating example.

Example 6.4. *Suppose node v_1 is a less important node with $c(v_1) = 0.0001$. In the i-th iteration, we find a node v_2 with marginal gain of 0.05. Due to the submodular property of the objective function, the marginal gain of v_1 must be smaller than*

$c(v_1) < 0.05$. *Thus we can safely prune* v_1 *from the* i*-th node selection without updating its marginal gain.*

Based on the motivating example, we come up with the *update when needed strategy*. The idea is that in the i-th iteration, we do not calculate the marginal gain for node $v \in V$ unless necessary. It is easy to adopt this strategy by modifying Algorithm 13. For each node $u \in V \setminus S$, $Score[u]$ stores its centrality or values calculated in the previous iteration, so does for the distance matrix M. In the i-th iteration, we first calculate the distance of nodes to the samples in P_i, then update the matrix and the score vector based on P_i. Next we sort the nodes decreasingly based on their values in $Score$. We continuously pop nodes from the queue and calculate their true marginal gains, until the largest marginal gain found is greater than the top value of the queue. Then we can safely prune all the untouched nodes from this iteration and select the node with the largest marginal gain as the next seed. Through this way, we can reduce the update cost for many unpromising nodes, which will greatly improve the efficiency when n is large.

6.4 Experiment

In this section, we present the results of a comprehensive performance study to evaluate the efficiency and effectiveness of the proposed techniques in this chapter.

6.4.1 Experiment Setup

In the experiments, we report the response time of finding k nodes to evaluate the efficiency of the algorithms. We also report the closeness centrality of the returned node set to measure the effectiveness.

Algorithms and Workload. We evaluate the performance of two proposed al-

gorithms, baseline sampling algorithm (**BSA**) and order based sampling algorithm (**OSA**). We omit the greedy algorithm in Algorithm 12, since it needs to compute and store all pairs shortest path distances. Even for a small graph with 50,000 nodes, it will need more than one day to return the results. We set the sample size for both BSA and OSA as 1000 to make a tradeoff between the accuracy and efficiency. In the experiments, we vary k from 1 to 50 with 50 by default. For each algorithm, we run 20 times and report the average performance.

Datasets. To demonstrate the effectiveness and efficiency of our methods, we conduct experiments on four real world social networks [2]. The parameters of the datasets are reported in Table 6.2. The diameter is defined as the longest shortest path distance in the graph. The datasets are available for download and further details about the datasets can also be referred.

Dataset	**n**	**m**	Diameter
Gowalla	196,591	950,327	14
Amazon	334,863	925,872	44
Youtube	1,134,890	2,987,624	20
LiveJournal	3,997,962	34,681,189	17

Table 6.2: Summary of Datasets

Implementation Environment. All experiments are carried out on a PC with Intel Xeon 2.30GHz and 96G RAM. The operating system is Redhat. All algorithms are implemented in C++ and compiled with GCC 4.8.2 with -O3 flag.

6.4.2 Efficiency Evaluation

In this section we evaluate the efficiency of the algorithms through response time. In the first set of experiments, Figure 6.3 reports the response time of BSA and

[2] `https://snap.stanford.edu/data/`

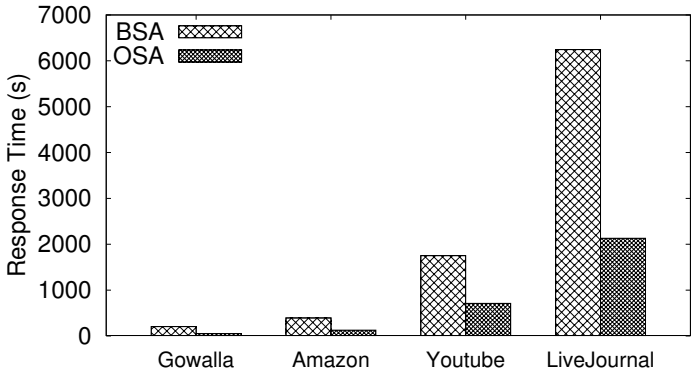
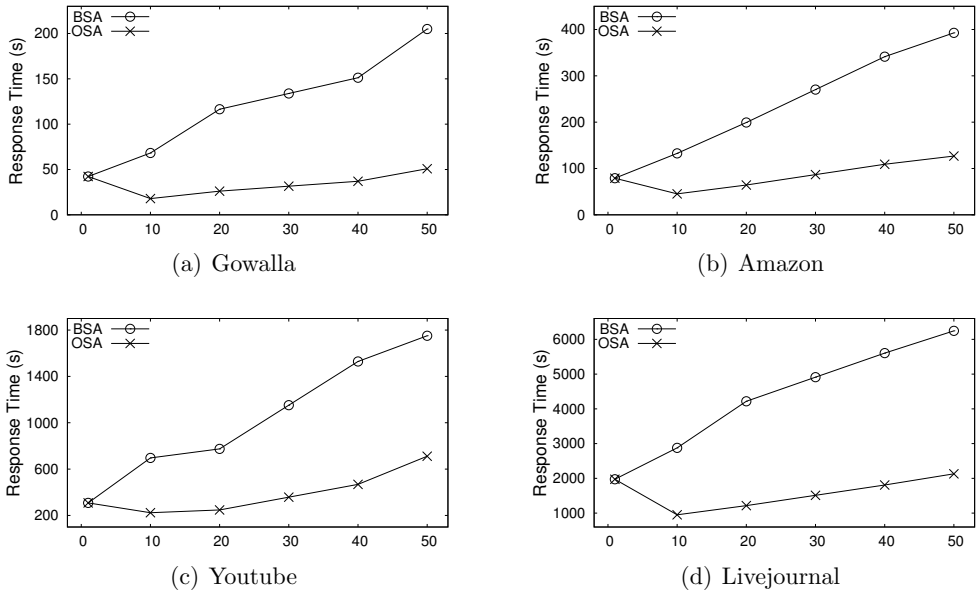


Figure 6.3: Efficiency Evaluation on All Datasets

OSA on all the datasets under the default settings. As the increase of dataset size, the response time grows for both algorithms. Because the cost of computing the distance from nodes to the samples and calculating the marginal gain will increase with the growing graph size. OSA constantly outperforms BSA by up to 4 times acceleration, due to the pruning in computing shortest path distances and updating the marginal gains.

In Figure 6.4, we report the response time by varying k from 1 to 50 on four datasets. When k equals 1, the response time of both algorithms is the same. It is because when k equals 1, there is no incremental node selection optimization and the updating optimization for OSA. Under the same sample size and $k = 1$, the procedure of OSA is the same as that of BSA. When k increases, the response time of BSA increases because of the increase of cost in the node selection. For OSA, the response time firstly drops then increases, because when k is larger than 1, OSA can take advantage of incremental node selection to reduce the cost of calculating shortest path distances to the samples. However, when k becomes larger, the node selection cost and marginal gain updating cost increase, which leads to the increase of the running time.



(a) Gowalla (b) Amazon

(c) Youtube (d) Livejournal

Figure 6.4: Efficiency Evaluation by Varying k

6.4.3 Effectiveness Evaluation

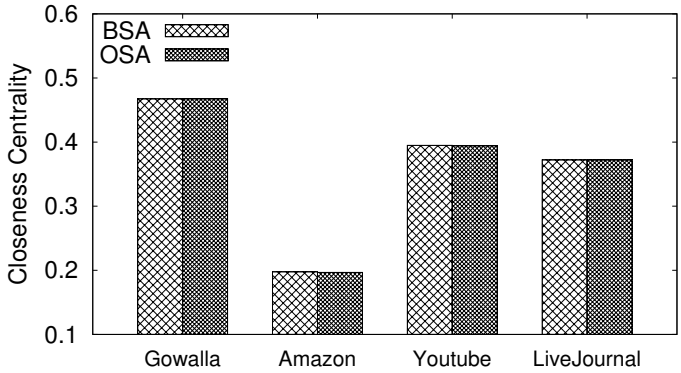
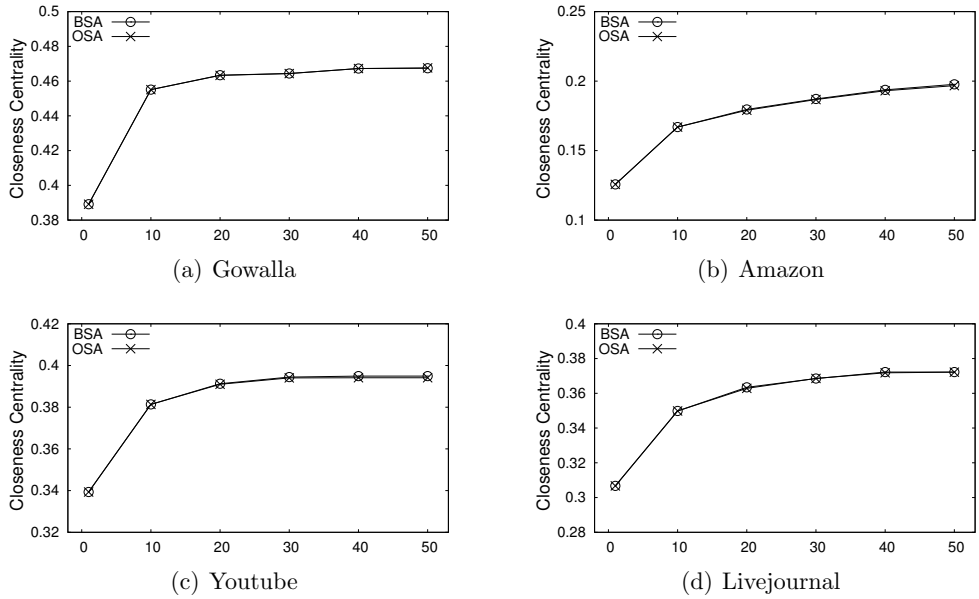


Figure 6.5: Effectiveness Evaluation on All Datasets

In this section, we evaluate the effectiveness of the proposed algorithms. In Figure 6.5, we report the closeness centrality on all the datasets under default settings. As can be seen, the quality of the results returned by OSA is almost the same as that of BSA. Only in few cases there is a very slight drop in the

closeness centrality returned by OSA. Since for each node selection in OSA, it only utilizes part of the samples for estimating, *i.e.*, $\cup_{j=1}^{i} P_j$ for selecting the i-th node. While BSA always uses the full samples to do the estimation. Also note that the closeness centrality of returned nodes is decided by the diameter of the graph. For graphs with small diameters, it tends to return a set of nodes with large closeness centrality when k is identical. In Figure 6.6, we report the closeness centrality by varying k from 1 to 50. When k increases, the closeness centrality increases for the returned node set. The quality difference of the results by both algorithms is very small. This is because the sample size should be proportional to k for the naïve sampling method in order to bound the quality of returned results. Therefore, by partitioning the sample into k parts and incrementally finding the node, it can offer similar quality results. For less dense graph with large diameters such as the Amazon dataset, the closeness centrality increases slowly when k is small (1 to 10).



(a) Gowalla (b) Amazon

(c) Youtube (d) Livejournal

Figure 6.6: Effectiveness Evaluation by Varying k

6.5 Conclusion

In this chapter, we consider closeness centrality for a set of nodes and aim to find the set with the largest closeness centrality. We show the problem is NP-Hard. By proving the monotonic and submodular properties of the objective function, we present a greedy framework which can achieve $1 - 1/e$ approximation ratio. Unfortunately, naïve implementation of the greedy framework will result in large space and time cost. To be able to scale to large graphs, we present two sampling based algorithms, BSA and OSA, respectively. OSA significantly accelerates BSA due to the optimizations in the shortest path distance computation and the updating procedure. By conducting extensive experiments on four real social networks, we demonstrate the efficiency and effectiveness of the proposed techniques.

Chapter 7

Final Remark

In this chapter, we provide a brief summarization of our research and describe some possible future directions. The major contributions of this thesis are concluded in Section 7.1. Section 7.2 introduces several possible orientations for future work.

7.1 Conclusions

Due to the information explosion, increasing number of existing techniques are facing the challenges of big data. To process massive datasets, a natural consideration is to use approximation methods. Unlike the parallel methods, sampling and sketching based approaches need much less resource and still can provide bounded solutions, which is acceptable for most applications. In this thesis, we apply the sampling and sketching techniques to tackle four problems where large datasets or high computation cost are involved. Below are the details.

On Gapped Set Intersection Size Estimation. Set intersection size estimation is a fundamental tool for data analysis and information retrieval. However, the gap information is neglected by existing work. In this thesis, we formally define the GSISE problem for both point query and range query. We extend the bottom-

k sketch to efficiently handle these queries. In addition, efficient algorithms are developed to conduct the estimation. Through carefully analysis, we derive the space required to bound the estimation quality. Moreover, we apply the techniques to solve top-K problem. Extensive experiments are conducted to evaluate the performance of proposed approaches.

Effective Order Preserving Estimation Method. Order preserving estimation is a very important tool in many applications, such as data visualization. To reduce the sample size required, we propose two effective methods, the interval separation method and the pairwise comparison method. In interval separation method, each group is considered individually, while in pairwise comparison method, we check the order of data by considering pairwise adjacent groups. To make the best use of samples, we dynamically allocate the input failure probability δ based on the current observed sample means. Finally, we conduct experiments on both real and synthetic datasets to demonstrate the effectiveness of the proposed techniques.

A Novel Scalable Method for Influence Maximization. Influence maximization is a key problem in viral marketing. As the state-of-the-art method has limitations in scalability, we propose an efficient solution, BKRIS, which combines the bottom-k sketch with the RIS framework. Specifically, we bring the order of samples into consideration, which makes it possible to achieve early termination before enumerating all the samples. To provide results with theoretical guarantees, we propose an efficient method to derive the lower bound of OPT. To handle the worst-case situation, several optimization techniques are developed. Finally, we conduct experiments on 10 real world social networks. Compared with the state-of-the-art method, we can achieve up to 2 orders of magnitude speedup.

Maximum Closeness Centrality Group Identification. We extend the concept of closeness centrality to a set of nodes and formally define the group closeness

centrality problem. We show the problem is NP-Hard, but the objective function is monotonic and submodular. Thus a greedy algorithm can achieve a result with $1 - 1/e$ approximation ratio. To scale to large graphs, we propose two sampling-based methods, BSA and OSA. OSA significantly accelerates the processing by incrementally selecting the nodes and using the pruning method to avoid computations for less important nodes. Our experiments on real social networks demonstrate the efficiency and effectiveness of the proposed methods.

7.2 Directions for Future Work

In this section, we propose several possible directions for future work.

7.2.1 GSISE Problem for Real Value Gap

As stated in Chapter 3, it is necessary to consider the gap information when conducting the set intersection size estimation. While the techniques proposed in this thesis can only solve the case where the input gap is integer type. However, in many applications, the input gap may be real values. Even though we can normalize all the real value gaps into integers, it may not be cost-effective for data storage. Thus, efficient and effective methods for processing real value gaps are required.

7.2.2 Optimal Sample Strategy for OPE Problem

The visualization of data order is a useful tool for many fields. In Chapter 4, we develop two effective stop functions by dynamically allocating the failure probability. However, the sample strategy used in this thesis is heuristic based. Even though the proposed methods are working well on both real and synthetic datasets, it is still attractive to develop the optimal sample strategy, which would have bound in

the worst case scenario.

7.2.3 Query-based Group Closeness Centrality

As we stated in Chapter 2, for influence maximization problem, there is a type of query-based settings. Thus it is natural to consider the query-based version for group closeness centrality maximization problem. For example, each node is associated with a weight decided by the input query. Naïvely, we can extend the uniform sampling method to a weighted sampling method by considering the weights of nodes. However, there might be a lot of queries issued, thus an index-based solution is required to answer each query efficiently. So the challenge is how to index sufficient samples or graph structures which can be utilized to answer any query with bounded guarantees.

Bibliography

[ABBB14] Çigdem Aslay, Nicola Barbieri, Francesco Bonchi, and Ricardo A. Baeza-Yates. Online topic-aware influence maximization queries. In *EDBT*, pages 295–306, 2014.

[ADHP09] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. Np-hardness of euclidean sum-of-squares clustering. *Mach. Learn.*, 75(2):245–248, 2009.

[AGW15] Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. Subcubic equivalences between graph centrality problems, APSP and diameter. In *SODA*, 2015.

[AMMIL12] Yaser S. Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning From Data*. AMLBook, 2012.

[AMS96] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *STOC*, pages 20–29, 1996.

[Bav48] A. Bavelas. A mathematical model for small group structures. *Human Organization*, 1948.

[Bav50] A. Bavelas. Communication patterns in task oriented groups. *Journal of the Acoustical Society of America*, 1950.

[BBC+16] Elisabetta Bergamini, Michele Borassi, Pierluigi Crescenzi, Andrea Marino, and Henning Meyerhenke. Computing top-k closeness centrality faster in unweighted graphs. In *ALENEX*, pages 68–80, 2016.

[BBCL14a] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. Maximizing social influence in nearly optimal time. In *SODA*, pages 946–957, 2014.

[BBCL14b] Christian Borgs, Michael Brautbar, Jennifer T. Chayes, and Brendan Lucier. Maximizing social influence in nearly optimal time. In *SODA*, pages 946–957, 2014.

[BBM12] Nicola Barbieri, Francesco Bonchi, and Giuseppe Manco. Topic-aware social influence propagation models. In *ICDM*, pages 81–90, 2012.

[BCFM98] Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. Min-wise independent permutations (extended abstract). In *Symposium on the Theory of Computing*, 1998.

[BHR+07] Kevin S. Beyer, Peter J. Haas, Berthold Reinwald, Yannis Sismanis, and Rainer Gemulla. On synopses for distinct-value estimation under multiset operations. In *SIGMOD Conference*, pages 199–210, 2007.

[BL15a] Phillip Bonacich and Paulette Lloyd. Eigenvector centrality and structural zeroes and ones: When is a neighbor not a neighbor? *Social Networks*, 43:86–90, 2015.

[BL15b] Yaron Singer Brendan Lucier, Joel Oren. Influence at scale: Distributed computation of complex contagion in networks. In *KDD*, 2015.

[BLL06] Jérémy Barbay, Alejandro López-Ortiz, and Tyler Lu. Faster adaptive set intersections for text searching. In *WEA*, 2006.

[BLLS09] Jérémy Barbay, Alejandro López-Ortiz, Tyler Lu, and Alejandro Salinger. An experimental investigation of set intersection algorithms for text searching. *Journal of Experimental Algorithmics*, 14, 2009.

[BM15] Rmi Bardenet and Odalric-Ambrym Maillard. Concentration inequalities for sampling without replacement. *Bernoulli*, 21(3):1361–1385, 2015.

[Bra01] Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25:163–177, 2001.

[Bro97] Andrei Z. Broder. On the resemblance and containment of documents. In *SEQUENCES97*, pages 21–29. IEEE Computer Society, 1997.

[CB02] G. Casella and R.L. Berger. *Statistical Inference*. Thomson Learning, 2002.

[CCK15] Shiri Chechik, Edith Cohen, and Haim Kaplan. Average distance queries through weighted samples in graphs and metric spaces: High scalability with tight statistical guarantees. In *APPROX/RANDOM*, 2015.

[CDN07] Surajit Chaudhuri, Gautam Das, and Vivek R. Narasayya. Optimized stratified sampling for approximate query processing. *TODS*, 32(2):9, 2007.

[CDPW14a] Edith Cohen, Daniel Delling, Thomas Pajor, and Renato F. Werneck. Computing classic closeness centrality, at scale. In *COSN*, pages 37–50, 2014.

[CDPW14b] Edith Cohen, Daniel Delling, Thomas Pajor, and Renato F. Werneck. Sketch-based influence maximization and computation: Scaling up with guarantees. In *CIKM*, pages 629–638, 2014.

[CFL+15] Shuo Chen, Ju Fan, Guoliang Li, Jianhua Feng, Kian-Lee Tan, and Jinhui Tang. Online topic-aware influence maximization. *PVLDB*, 2015.

[CGHJ12] Graham Cormode, Minos N. Garofalakis, Peter J. Haas, and Chris Jermaine. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends in Databases*, 4(1-3):1–294, 2012.

[Che52] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):493–507, 1952.

[CK07] Edith Cohen and Haim Kaplan. Summarizing data using bottom-k sketches. In *PODC*, pages 225–234, 2007.

[CK08] Edith Cohen and Haim Kaplan. Tighter estimation using bottom k sketches. *PVLDB*, 1(1):213–224, 2008.

[CK09] Edith Cohen and Haim Kaplan. Leveraging discarded samples for tighter estimation of multiple-set aggregates. In *SIGMETRICS/Performance*, 2009.

[CM04] Graham Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. In *LATIN*, pages 29–38, 2004.

[CM10] J. Shane Culpepper and Alistair Moffat. Efficient set intersection for inverted indexing. *ACM Trans. Inf. Syst.*, 29(1):1, 2010.

[Coh97] Edith Cohen. Size-estimation framework with applications to transitive closure and reachability. *J. Comput. Syst. Sci.*, 55(3):441–453, 1997.

[Coh00] Edith Cohen. Polylog-time and near-linear work approximation scheme for undirected shortest paths. *J. ACM*, 47(1):132–166, 2000.

[Coh14] Edith Cohen. All-distances sketches, revisited: HIP estimators for massive graphs analysis. In *PODS*, pages 88–99, 2014.

[CSRL01] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[CWW10] Wei Chen, Chi Wang, and Yajun Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *SIGKDD*, pages 1029–1038, 2010.

[CWY09] Wei Chen, Yajun Wang, and Siyu Yang. Efficient influence maximization in social networks. In *KDD*, pages 199–208, 2009.

[CYZ10] Wei Chen, Yifei Yuan, and Li Zhang. Scalable influence maximization in social networks under the linear threshold model. In *ICDM*, pages 88–97, 2010.

[Dat09] DataExpo. Flight records. `http://stat-computing.org/dataexpo/2009/the-data.html`, 2009.

[DGH52] D. J. Thompson D. G. Horvitz. A generalization of sampling without replacement from a finite universe. *Journal of the American Statistical Association*, 47(260):663–685, 1952.

[DK11] Bolin Ding and Arnd Christian König. Fast set intersection in memory. *PVLDB*, 4(4):255–266, 2011.

[DLM00] Erik D. Demaine, Alejandro López-Ortiz, and J. Ian Munro. Adaptive set intersections, unions, and differences. In *SODA*, 2000.

[DR96] Devdatt Dubhashi and Desh Ranjan. Balls and bins: A study in negative dependence. *Random Structures and Algorithms*, 13:99–124, 1996.

[DR01] Pedro Domingos and Matt Richardson. Mining the network value of customers. In *KDD*, pages 57–66, 2001.

[EW01] David Eppstein and Joseph Wang. Fast approximation of centrality. In *SODA*, 2001.

[Fre78] Linton C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, page 215, 1978.

[FY38] R. A. Fisher and F. Yates. Statistical tables for biological, agricultural and medical research, 1938.

[GBS11] Manuel Gomez-Rodriguez, David Balduzzi, and Bernhard Schölkopf. Uncovering the temporal dynamics of diffusion networks. In *ICML*, pages 561–568, 2011.

[GLL11a] Amit Goyal, Wei Lu, and Laks V. S. Lakshmanan. SIMPATH: an efficient algorithm for influence maximization under the linear threshold model. In *ICDM*, pages 211–220, 2011.

[GLL11b] Amit Goyal, Wei Lu, and Laks V.S. Lakshmanan. Celf++: Optimizing the greedy algorithm for influence maximization in social networks. In *WWW*, pages 47–48, 2011.

[GP06] Sreenivas Gollapudi and Rina Panigrahy. Exploiting asymmetry in hierarchical topic extraction. In *CIKM*, pages 475–482, 2006.

[HC05] Jiang-Liang Hou and Chuan-An Chan. Method for keyword correlation analysis. US Patent 20050071365 A1, 2005.

[HGI00] Taher Haveliwala, Aristides Gionis, and Piotr Indyk. Scalable techniques for clustering the web (extended abstract). In *Third International Workshop on the Web and Databases (WebDB 2000)*, 2000.

[HL72] Frank K. Hwang and Shen Lin. A simple algorithm for merging two disjoint linearly-ordered sets. *SIAM J. Comput.*, 1(1), 1972.

[Hoa61] C. A. R. Hoare. Algorithm 65: Find. *Commun. ACM*, 4(7):321–322, July 1961.

[Hoe62] Wassily Hoeffding. Probability inequalities for sums of bounded random variables, 1962.

[HS92] Peter J. Haas and Arun N. Swami. Sequential sampling procedures for query size estimation. In *SIGMOD*, pages 341–350, 1992.

[I+98] Piotr Indyk et al. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, 1998.

[IBS08] Ihab F. Ilyas, George Beskales, and Mohamed A. Soliman. A survey of top-k query processing techniques in relational database systems. *ACM Comput. Surv.*, 40(4), 2008.

[Ind99] Piotr Indyk. Sublinear time algorithms for metric space problems. In *STOC*, 1999.

[Iof10] Sergey Ioffe. Improved consistent sampling, weighted minhash and l1 sketching. In *ICDM*, pages 246–255, 2010.

[Joh77] Donald B. Johnson. Efficient algorithms for shortest paths in sparse networks. *J. ACM*, 24(1):1–13, 1977.

[KBP+15] Albert Kim, Eric Blais, Aditya G. Parameswaran, Piotr Indyk, Samuel Madden, and Ronitt Rubinfeld. Rapid sampling for visualizations with ordering guarantees. *PVLDB*, 8(5):521–532, 2015.

[KKT03] David Kempe, Jon M. Kleinberg, and Eva Tardos. Maximizing the spread of influence through a social network. In *SIGKDD*, pages 137–146, 2003.

[KS06] Masahiro Kimura and Kazumi Saito. Tractable models for information diffusion in social networks. In *PKDD*, pages 259–271, 2006.

[LCF+14] Guoliang Li, Shuo Chen, Jianhua Feng, Kian-Lee Tan, and Wen-Syan Li. Efficient location-aware influence maximization. In *SIGMOD 2014*, pages 87–98, 2014.

[LK10] Ping Li and Arnd Christian König. b-bit minwise hashing. In *WWW*, 2010.

[LKG+07] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *KDD*, pages 420–429, 2007.

[LZT15] Yuchen Li, Dongxiang Zhang, and Kian-Lee Tan. Real-time targeted influence maximization for online advertisements. *PVLDB*, 2015.

[McD89] C. McDiarmid. On the method of bounded differences. In *Surveys in Combinatorics*, number 141 in London Mathematical Society Lecture Note Series, pages 148–188. Cambridge University Press, 1989.

[MMT07] Mark Manasse, Frank McSherry, and Kunal Talwar. Consistent weighted sampling. *Unpublished technical report) http://research. microsoft. com/en-us/people/manasse*, 2007.

[MPP14] Michael Mitzenmacher, Rasmus Pagh, and Ninh Pham. Efficient estimation for high similarities using odd sketches. In *WWW*, 2014.

[Ney34] Jerzy Neyman. On the two different aspects of the representative method: The method of stratified sampling and the method of purposive selection. *Journal of the Royal Statistical Society*, 97(4):558–625, 1934.

[Ney37] J. Neyman. Outline of a theory of statistical estimation based on the classical theory of probability. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 236(767):333–380, 1937.

[NWF78] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Math. Program.*, 14(1):265–294, 1978.

[OCL08] Kazuya Okamoto, Wei Chen, and Xiang-Yang Li. Ranking of closeness centrality for large-scale social networks. In *FAW*, 2008.

[OLH14] Paul W. Olsen, Alan G. Labouseur, and Jeong-Hyon Hwang. Efficient top-k closeness centrality search. In *ICDE*, 2014.

[PSW14] Rasmus Pagh, Morten Stöckel, and David P. Woodruff. Is min-wise hashing optimal for summarizing set intersection? In *PODS*, 2014.

[RD02] Matthew Richardson and Pedro Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD*, pages 61–70, 2002.

[RMJ07] Matthew J. Rattigan, Marc E. Maier, and David Jensen. Graph clustering with network structure indices. In *ICML*, 2007.

[SEMP14] Moritz Sudhof, Andrés Goméz Emilsson, Andrew L. Maas, and Christopher Potts. Sentiment expression conditioned by affective transitions and social forces. In *SIGKDD*, 2014.

[Ser74] R. J. Serfling. Probability inequalities for the sum in sampling without replacement. *The Annals of Statistics*, 2(1):39–48, 1974.

[SWL11] Benjamin Schlegel, Thomas Willhalm, and Wolfgang Lehner. Fast sorted-set intersection using SIMD instructions. In *ADMS*, 2011.

[SWQ+14] Yifang Sun, Wei Wang, Jianbin Qin, Ying Zhang, and Xuemin Lin. SRS: solving c-approximate nearest neighbor queries in high dimensional euclidean space with a tiny index. *PVLDB*, 8(1):1–12, 2014.

[Tho01] Mikkel Thorup. Quick k-median, k-center, and facility location for sparse graphs. In *ICALP*, 2001.

[TKC+14] Manuel Then, Moritz Kaufmann, Fernando Chirigati, Tuan-Anh Hoang-Vu, Kien Pham, Alfons Kemper, Thomas Neumann, and Huy T. Vo. The more the merrier: Efficient multi-source graph traversal. *PVLDB*, 8(4):449–460, 2014.

[TSX15] Youze Tang, Yanchen Shi, and Xiaokui Xiao. Influence maximization in near-linear time: A martingale approach. In *SIGMOD*, pages 1539–1554, 2015.

[TXS14] Youze Tang, Xiaokui Xiao, and Yanchen Shi. Influence maximization: near-optimal time complexity meets practical efficiency. In *SIGMOD*, pages 75–86, 2014.

[WC14] Cheng Yang Wei Chen, Tian Lin. Real-time topic-aware influence maximization using preprocessing. In *Arxiv.org*, 2014.

[WJ09] Mingxi Wu and Chris Jermaine. Guessing the extreme values in a data set: a bayesian method and its applications. *VLDB J.*, 18(2):571–597, 2009.

[WKF94] S. Wasserman and editors K. Faust. Social network analysis: Methods and applications. *Cambridge University Press*, 1994.

[WZZL16] Xiaoyang Wang, Ying Zhang, Wenjie Zhang, and Xuemin Lin. Distance-aware influence maximization in geo-social network. In *ICDE*, pages 1–12, 2016.

[ZLG11] Jiaqi Zhai, Yin Lou, and Johannes Gehrke. Atlas: a probabilistic algorithm for high dimensional similarity search. In *SIGMOD Conference*, pages 997–1008, 2011.

[ZLTG14] Junzhou Zhao, John C. S. Lui, Don Towsley, and Xiaohong Guan. Measuring and maximizing group closeness centrality over disk-resident graphs. In *WWW '14 Companion Volume*, pages 689–694, 2014.

[ZPC+15] Wen-Yuan Zhu, Wen-Chih Peng, Ling-Jyh Chen, Kai Zheng, and Xiaofang Zhou. Modeling user mobility for location promotion in location-based social networks. In *KDD*, 2015.