Dynamic Covering for Recommendation Systems

Ioannis Antonellis
oDesk
antonell@cs.stanford.edu

Anish Das Sarma
Google Research
anish.dassarma@gmail.com

Shaddin Dughmi
Microsoft Research
shaddin@gmail.com

ABSTRACT

In this paper, we identify a fundamental algorithmic problem that we term *succinct dynamic covering* (SDC), arising in many modern-day web applications, including ad-serving and online recommendation systems such as in eBay, Netflix, and Amazon. Roughly speaking, SDC applies two restrictions to the well-studied Max-Coverage problem [14]: Given an integer k, $\mathcal{X} = \{1, 2, \ldots, n\}$ and $\mathcal{I} = \{S_1, \ldots, S_m\}$, $S_i \subseteq \mathcal{X}$, find $\mathcal{J} \subseteq \mathcal{I}$, such that $|\mathcal{J}| \leq k$ and $(\bigcup_{S \in \mathcal{J}} S)$ is as large as possible. The two restrictions applied by SDC are: (1) *Dynamic:* At query-time, we are given a *query* $Q \subseteq \mathcal{X}$, and our goal is to find $\mathcal{J}$ such that $Q \bigcap (\bigcup_{S \in \mathcal{J}} S)$ is as large as possible; (2) *Space-constrained:* We don't have enough space to store (and process) the entire input; specifically, we have $o(mn)$, and maybe as little as $O((m+n)polylog(mn))$ space. A solution to SDC maintains a small data structure, and uses this datastructure to answer *most* dynamic queries with high accuracy. We call such a scheme a *Coverage Oracle*.

We present algorithms and complexity results for coverage oracles. We present deterministic and probabilistic near-tight upper and lower bounds on the approximation ratio of SDC as a function of the amount of space available to the oracle. Our lower bound results show that to obtain constant-factor approximations we need $\Omega(mn)$ space. Fortunately, our upper bounds present an explicit tradeoff between space and approximation ratio, allowing us to determine the amount of space needed to guarantee certain accuracy.

Categories and Subject Descriptors

H.0 [**Information Systems**]: General

General Terms

Algorithms

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.
Copyright 2012 ACM 978-1-4503-1156-4/12/10 ...$15.00.

Keywords

dynamic covering, recommendation systems, max-coverage problem

1. INTRODUCTION

The explosion of data and applications on the web over the last decade have given rise to many new data management challenges. This paper identifies a fundamental subproblem inherent in several Web applications, including online recommendation systems, and serving advertisements on webpages. Let us begin with a motivating example.

Example 1.1. *Consider the online movie rental and streaming website, Netflix [3], and one of their users Alice. Based on Alice's movie viewing (and rating) history, Netflix would like to recommend new movies to Alice for watching. (Indeed, Netflix threw open a million-dollar challenge on significantly improving their movie recommendations [4].) Conceivably, there are many ways of devising algorithms for recommendation ranging from data mining to machine learning techniques, and indeed there has been a great deal of such work on providing personalized recommendations (see [6] for a survey). Regardless of the specific technique, an important subproblem that arises is finding users "similar" to Alice, i.e., finding users who have independently or in conjunction viewed (and liked) movies seen (and liked) by Alice.*

Abstractly speaking, we are given a universal set *of all Netflix movies, and Netflix users identified by the* subset *of movies they have viewed (and liked or disliked). Given a specific user Alice, we are interested in finding (say k) other users, who together* cover *a large set of Alice's likes and dislikes. Note that for each user, the set of movies that need to be covered is different, and therefore the covering cannot be performed* statically, *independent of the user. In fact, Netflix* dynamically *provides movie recommendations as users rate movies in a particular genre (say comedy), or request movies in specific languages, or time periods. Providing recommendations at interactive speed, based on user queries (such as a particular genre), rules out computationally-expensive processing over the entire Netflix data, which is very large.*[1] *Therefore, we are interested in* approximately *solving the aforementioned covering problem based on a* subset *of the data.*

[1] Netflix currently has over 10 millions users, over 100,000 movies, and obviously some of the popular movies have been viewed by many users, and movie buffs have rated a large number of movies; Netflix owns over 55 million discs.

The main challenge that arises is to statically *identify a* subset *of the data that would provide good approximations to the covering problem for any* dynamic *user query.*

Note that very similar challenges arise in other recommendation systems, e.g., Alice visits an online shopping website like eBay [2] or Amazon [1], and the website is interested in recommending products to her based on her current query for a particular brand or product, and her prior purchasing (and viewing) history.

The example above can be formulated as an instance of a simple algorithmic covering problem, generalizing the NP-hard optimization problem *max k-cover* [14]. The input to this problem is an integer k, a set $\mathcal{X} = \{1, \ldots, n\}$, a family $\mathcal{I} \subseteq 2^{\mathcal{X}}$ of subsets of $\mathcal{X}$, and *query* $Q \subseteq X$. Here $(\mathcal{X}, \mathcal{I})$ is called a *set system*, $\mathcal{X}$ is called the *ground set* of the set-system, and members of $\mathcal{X}$ are called *elements* or *items*. We make no assumptions on how the set system is represented in the input, though the reader can think of the obvious representation by a $n \times m$ bipartite graph for intuition. This $n \times m$ bipartite graph can be stored in $O(nm)$ bits, which is in fact information-theoretically optimal for storing an arbitrary set system on n items and m sets. The objective of the problem is to return $\mathcal{J} \subseteq \mathcal{I}$ with $|\mathcal{J}| \leq k$ that collectively cover as much of Q as possible. Since this problem is a generalization of max k-cover, it is NP-hard. Nevertheless, absent any additional constraints this problem can be approximated in polynomial time by a straightforward adaptation of the greedy algorithm for max k-cover [2], which attains a constant factor $\frac{e}{e-1}$ approximation in $O(mn)$ time [16]. However, we further constrain the problem as follows, rendering new techniques necessary.

From the above example, we identify two properties that we require of any system that solves this covering problem:

1. **Space Constrained:** We need to (statically) preprocess the set system $(\mathcal{X}, \mathcal{I})$ and store a small sketch (much smaller than $O(mn)$), in the form of a *data structure*, and discard the original representation of $(\mathcal{X}, \mathcal{I})$. This can be thought of as a form of lossy compression. We do not require the data structure to take any particular form; it need only be a sequence of bits that allows us to extract information about the original set system $(\mathcal{X}, \mathcal{I})$. For instance, any statistical summary, a subgraph of the bipartite graph representing the set system, or other representation is acceptable.

2. **Dynamic**: The query Q is not known a-priori, but arrives *dynamically*. More precisely: Q arrives *after* the data structure is constructed and the original data discarded. It is at that point that the data structure must be used to compute a solution $\mathcal{J}$ to the covering problem.

We call this covering problem (formalized in the next section) the *Succinct Dynamic Covering* (SDC) problem. Moreover, we call a solution to SDC a *Coverage Oracle*. A coverage oracle consists of a *static stage* that constructs a datastructure, and a *dynamic stage* that uses the datastructure to answer queries.

Next we briefly present another, entirely different, Web application that also needs to confront SDC . In addition, we note that there are several other applications facing similar covering problems, including gene identification [13], searching domain-specific aggregator sites like Yelp [5], topical query decomposition [9], and search-result diversification [10, 12].

Example 1.2. *Online advertisers bid on (1) webpages matching relevancy criteria and (2) typically target a certain user demographic. Advertisements are served based on a combination of the two criterion above. When a user visits a particular webpage, there is usually no precise information about the users' demographic, i.e., age, location, interests, gender, etc. Instead, there is a* range *of possible values for each of these attributes, determined based on the search query the user issued or session information. Ad-servers therefore attempt to pick a set of advertisements that would be of interest (i.e., "cover") a large number of users; the user demographic that needs to be covered is determined by the page on which the advertisement is being placed, the user query, and session information. Therefore, ad-serving is faced with the SDC problem. The space constraint arises because the set system consisting of all webpages, and each user identified by the set of webpages visited by the user is prohibitively large to store in memory and process in real-time for every single page view. The dynamic aspect arises because each user view of each page is associated with a different user demographic that needs to be covered.*

1.1 Contributions and Outline

Next we outline the main contributions of this paper.

- In Section 3 we formally define the succinct dynamic covering (SDC) problem, and summarize our results.
- In Section 4 we present a randomized coverage oracle for SDC . The oracle is presented as a function of the available space, thus allowing us to tradeoff space for accuracy based on the specific application. Unfortunately, the approximation ratio of this oracle degrades rapidly as space decreases; However, the next section shows that this is in fact unavoidable.
- In Section 5 we present a lowerbound on the best possible approximation attainable as a function of the space allowed for the datastructure. This lowerbound essentially matches the upperbound of Section 4, though with the caveat that the lowerbound is for oracles that do not use randomization. We expect the lowerbound to hold more generally for randomized oracles, though we leave this as an open question.

Related work is presented next, and future directions are presented in Section 6. To maintain the flow of the paper, we defer some of the longer proofs of our results to Appendix A, with brief sketches appearing in the main body of the paper.

2. RELATED WORK

Our study of the tradeoff between space and approximation ratio is in the spirit of the work of Thorup and Zwick [19] on *distance oracles*. They considered the problem of compressing a graph G into a small datastructure, in such a way that the datastructure can be used to approximately

[2] The greedy algorithm for max k-cover, adapted to our problem, is simple: Find the set in $\mathcal{I}$ covering as many uncovered items in Q as possible, and repeat this k times. This can clearly be implemented in $O(mn)$ time, and has been shown to yield a $e/(e-1)$ approximation.

answer queries for the distance between pairs of nodes in G. Similar to our results, they showed matching upper and lower bounds on the space needed for compressing the graph subject to preserving a certain approximation ratio. Moreover, similarly to our upperbounds for SDC, their distance oracles benefit from a speedup at query time as approximation ratio is sacrificed for space.

Previous work has studied the set cover problem under streaming models. One model studied in [8, 15] assumes that the sets are known in advance, only elements arrive online, and, the algorithms do not know in advance which subset of elements will arrive. An alternative model assumes that elements are known in advance and sets arrive in a streaming fashion [18]. Our work differs from these works in that SDC operates under a storage budget, so all sets cannot be stored; moreover, SDC needs to provide a good cover for *all* possible dynamic query inputs.

Another related area is that of nearest neighbor search. It is easy to see that the SDC problem with $k = 1$ corresponds to nearest neighbor search using the dot product similarity measure, i.e., $sim_{dot}(x, y) = \frac{dot(x,y)}{n}$. However, following from a result from Charikar [11], there exists no locality sensitive hash function family for the dot product similarity function. Thus, there is no hope that signature schemes (like minhashing for the Jaccard distance) can be used for SDC .

3. SDC

We start by defining the succinct dynamic covering (SDC) problem in Section 3.1. Then, in Section 3.2 we summarize the main technical results achieved by this paper.

3.1 Problem Definition

We now formally define the SDC problem.

Definition 3.1 (SDC). *Given an offline input consisting of a set system* $(\mathcal{X}, \mathcal{I})$ *with* n elements *(a.k.a* items*)* $\mathcal{X}$ *and* m sets $\mathcal{I}$, *and an integer* $k \geq 1$, *devise a* coverage oracle *such that given a dynamic* query $Q \subseteq \mathcal{X}$, *the oracle finds a* $\mathcal{J} \subseteq \mathcal{I}$ *such that* $|\mathcal{J}| \leq k$ *and* $(\bigcup_{S \in \mathcal{J}} S) \bigcap Q$ *is as large as possible.*

Definition 3.2 (Coverage Oracle). *A* Coverage Oracle *for SDC consists of two stages:*

1. **Static Stage:** *Given integers m,n,k, and set system* $(\mathcal{X}, \mathcal{I})$ *with* $|\mathcal{X}| = n$ *and* $|\mathcal{I}| = m$, *build a datastructure* $\mathcal{D}$.
2. **Dynamic Stage:** *Given a dynamic* query $Q \subseteq \mathcal{X}$, *use* $\mathcal{D}$ *to return* $\mathcal{J} \subseteq \mathcal{I}$ *with* $|\mathcal{J}| \leq k$ *as a solution to SDC.*

Note that our two constraints on a solution for SDC are illustrated by the two stages above. (1) We are interested in building an offline data structure $\mathcal{D}$, and only use $\mathcal{D}$ to answer queries. Typically, we want to maintain a *small* data structure, certainly $o(mn)$, and maybe as little as $O((m + n)polylog(mn))$ or even $O(m + n)$. Therefore, we cannot store the entire set system. (2) Unlike the traditional max-coverage problem where the entire set of elements $\mathcal{X}$ need to be covered, in SDC we are given queries dynamically. Therefore, we want a coverage oracle that returns good solutions for all queries.

Given the space limitation of SDC, we cannot hope to exactly solve SDC (for all dynamic input queries). The goal of this paper is to explore *approximate* solutions for SDC, given a specific space constraint on the offline data structure $\mathcal{D}$. We define the *approximation ratio* of an oracle as the worst-case, taken over all inputs, of the ratio between the coverage of Q by the optimal solution and the coverage of Q by the output of the oracle. We allow the approximation ratio to be a function of n, m, and k, and denote it by $\alpha(n, m, k)$.

More precisely, given a coverage oracle $\mathcal{A}$, if on inputs $k, \mathcal{X}, \mathcal{I}, Q$ (where implicitly $n = |\mathcal{X}|$ and $m = |\mathcal{I}|$) the oracle $\mathcal{A}$ returns $\mathcal{J} \subseteq \mathcal{I}$, we denote the size of the coverage as $\mathcal{A}(k, \mathcal{X}, \mathcal{I}, Q) := |(\bigcup_{S \in \mathcal{J}} S) \bigcap Q|$. Similarly, we denote the coverage of the optimal solution by $OPT(k, \mathcal{X}, \mathcal{I}, Q) := \max\{|(\bigcup_{S \in \mathcal{J}^*} S) \bigcap Q| : \mathcal{J}^* \subseteq \mathcal{I}, |\mathcal{J}^*| \leq k\}$. We then express the *approximation ratio* $\alpha(n, m, k)$ as follows.

$$\alpha(n, m, k) = \max \frac{OPT(k, \mathcal{X}, \mathcal{I}, Q)}{\mathcal{A}(k, \mathcal{X}, \mathcal{I}, Q)}$$

Where the maximum above is taken over set systems $(\mathcal{X}, \mathcal{I})$ with $|\mathcal{X}| = n$ and $|\mathcal{I}| = m$, and queries $Q \subseteq \mathcal{X}$.

We will also be concerned with *randomized* coverage oracles. Note that, when we devise randomized coverage oracle, we use randomization only in the static stage; i.e. in the construction of the datastructure. We then let the *expected approximation ratio* be the worst case *expected* performance of the oracle as compared to the optimal solution.

$$\alpha(n, m, k) = \max \left(\mathbf{E} \left[\frac{OPT(k, \mathcal{X}, \mathcal{I}, Q)}{\mathcal{A}(k, \mathcal{X}, \mathcal{I}, Q)} \right] \right) \tag{1}$$

The expectation in the above expression is over the random coins flipped by the static stage of the oracle, and the maximization is over $\mathcal{X}, \mathcal{I}, Q$ as before. We elaborate on this benchmark in Section 4.

We study the space-approximation tradeoff; i.e., how the (expected) approximation ratio improves as the amount of space allowed for $\mathcal{D}$ is increased. In our lowerbounds, we are not specifically concerned with the time taken to compute the datastructure or answer queries. Therefore, our lowerbounds are purely *information-theoretic*: we calculate the amount of information we are required to store if we are to guarantee a specific approximation ratio, independent of computational concerns. Our lowerbounds are particularly novel and striking in that *they assume nothing* about the datastructure, which may be an arbitrary sequence of bits. We establish our lowerbounds via a novel application of the probabilistic method that may be of independent interest.

Even though we focus on space vs approximation, and not on runtime, fortunately the coverage oracles in our upperbounds can be implemented efficiently (both static and dynamic stage). Moreover, using our upperbounds to trade approximation for space yields, as a side-effect, an improvement in runtime when answering a query. In particular, observe that if no sparsification of the data is done up-front, then answering each query using the standard greedy approximation algorithm for max k-cover [16] takes $O(mn)$ time. Our oracles, presented in Section 4, spends $O(mn)$ time up-front building a data structure of size $O(b)$, where b is a parameter of the oracle between n and nm. In the dynamic stage, however, answering a query now takes $O(b)$, since we use the greedy algorithm for max k-cover on a "sparse" set system. Therefore, the dynamic stage becomes

Table 1: Summary of SDC results giving the approximation-ratio, space constraint on coverage oracle, and whether the nature of the bound: upper bound (UB) or lower bound (LB) and deterministic (Det.) or randomized (Rand.)

Approximation Ratio	Storage	Bound
$O\left(\min\left(\frac{m}{k}, \sqrt{\frac{n}{k}}\right)\right)$	$\widetilde{O}(n)$	Det. UB
$O\left(\min\left(\frac{m^\epsilon}{\sqrt{k}}, \sqrt{\frac{n}{k}}\right)\right)$	$\widetilde{O}(nm^{1-2\epsilon})$	Rand. UB
$\Omega\left(\min\left(\frac{m^{\epsilon-\delta_1}}{k\sqrt{k}}, \frac{n^{1/2-\delta_2}}{k\sqrt{k}}\right)\right)$	$\widetilde{O}(nm^{1-2\epsilon})$	Det. LB

faster as we decrease size of the data structure. In fact, this increase in speed is not restricted to an algorithmic speedup as described above. It is likely that there will also be speedup due to architectural reasons, since a smaller amount of data needs to be kept in memory. Therefore, trading off approximation for space yields an incidental speedup in runtime which bodes well for the dynamic nature of the queries.

3.2 Summary of results

Table 1 summarizes the main results obtained in this paper for SDC input with n elements, m sets, and integer $k \geq 1$. The lower bound in the table is for any nonnegative constants δ_1, δ_2 not both 0, and the randomized upperbound is parameterized by ϵ with $0 \leq \epsilon \leq 1/2$. The upper and lower bounds are developed in Sections 4 and 5 respectively.

4. UPPER BOUNDS

In this section, we show a coverage oracle that trades off space and approximation ratio. We designate a tradeoff parameter ϵ, where $0 \leq \epsilon \leq 1/2$. For any such ϵ, we get an $O\left(\frac{\min(m^\epsilon, \sqrt{n})}{\sqrt{k}}\right)$-approximate coverage oracle that stores $\widetilde{O}(nm^{1-2\epsilon})$ bits. Therefore, setting a small value of ϵ achieves a better approximation ratio, at the expense of storage space. As is common practice, we use $\widetilde{O}()$ to denote suppressing polylogarithmic factors in n and m; this is reasonable when the guarantees are super-polylogarithmic, as is the case here.

The oracle we show is randomized, in the sense that the static stage flips some random coins. The datastructure constructed is a random variable in the internal coin flips of the static stage of the oracle. We measure the *expected approximation ratio* (a.k.a approximation ratio, when clear from context) of the oracle, as defined in Equation (1). For every fixed query Q independent of the random coins used in constructing the datastructure, this ratio is attained in expectation. In other words, our adversarial model is that of an *oblivious adversary*: someone trying to fool our oracle may choose any query they like, but their choice cannot depend on knowledge of the random choices made in constructing the datastructure.

In Section 5 we will see that our oracle attains a space-approximation tradeoff that is essentially optimal when compared with oracles that are deterministic. In other words, no deterministic oracle can do substantially better. We leave open the questions of whether a better randomized oracle is possible, and whether an equally good deterministic oracle exists.

4.1 Main Result and Roadmap

The following theorem states the main result of this section.

THEOREM 4.1. *For every ϵ with $0 \leq \epsilon \leq 1/2$, there is a randomized coverage oracle for SDC that achieves an $O\left(\frac{\min(m^\epsilon, \sqrt{n})}{\sqrt{k}}\right)$ approximation and stores $\widetilde{O}(nm^{1-2\epsilon})$ bits.*

The remainder of this section, leading up to the above result, is organized as follows. Before proving Theorem 4.1, to build intuition we show in Section 4.2 (Remark 4.2) a much simpler deterministic oracle, with a much weaker approximation guarantee. Then, we prove Theorem 4.1 in two parts. First, in Section 4.3, we show a randomized coverage oracle that stores $\widetilde{O}(nm^{1-2\epsilon})$ bits and achieves an $O(m^\epsilon/\sqrt{k})$ approximation in expectation. Then, in Section 4.4, we show a deterministic oracle that achieves a $O(\sqrt{n}/\sqrt{k})$ approximation and stores $\widetilde{O}(n)$ bits. Combining the two oracles into one in the obvious way yields Theorem 4.1.

4.2 Simple Deterministic Oracle

REMARK 4.2. *There is a simple deterministic oracle that attains a m/k approximation with $\widetilde{O}(n)$ space. The static stage proceeds as follows: Given set system $(\mathcal{X}, \mathcal{I})$, for each $i \in \mathcal{X}$ we "remember" one set $S \in \mathcal{I}$ with $i \in S$ (breaking ties arbitrarily). In other words, for each $S \in \mathcal{I}$ we define $\widehat{S} \subseteq S$ such that $\left\{\widehat{S} : S \in \mathcal{I}\right\}$ is a partition of $\mathcal{X}$. We then store the "sparsified" set system $\left(\mathcal{X}, \widehat{\mathcal{I}} = \left\{\widehat{S} : S \in \mathcal{I}\right\}\right)$. It is clear that this can be done in linear time by a trivial greedy algorithm. Moreover, $(\mathcal{X}, \widehat{\mathcal{I}})$ can be stored in $\widetilde{O}(n)$ space as a $n \times m$ bipartite graph with n edges.*

The dynamic stage is straightforward: given a query Q, we simply return the indices of the k sets in $\widehat{\mathcal{I}}$ that collectively cover as much of Q as possible. It is clear that this gives a m/k approximation. Moreover, since $\widehat{\mathcal{I}}$ is a partition of $\mathcal{X}$, it can be accomplished by a trivial greedy algorithm in polynomial time.

Next we use randomization to show a much better, and much more involved, upperbound that trades off approximation and space.

4.3 An $O(m^\epsilon/\sqrt{k})$ Approximation with $\widetilde{O}(nm^{1-2\epsilon})$ Space

Consider the set system $(\mathcal{X}, \mathcal{I})$, where $\mathcal{X}$ is the set of items and $\mathcal{I}$ is the family of sets. We assume without loss that each item is in some set. We define a randomized oracle for building a datastructure, which is a "sparsified" version of $(\mathcal{X}, \mathcal{I})$. Namely, for every $S \in \mathcal{I}$ we define $\widehat{S} \subseteq S$, and store the set system $\left(\mathcal{X}, \widehat{\mathcal{I}} = \left\{\widehat{S}\right\}_{S \in \mathcal{I}}\right)$. We require that $(\mathcal{X}, \widehat{\mathcal{I}})$ can be stored in $\widetilde{O}(nm^{1-2\epsilon})$ space. We construct the datastructure in two stages, as follows.

- Label all items in $\mathcal{X}$ "uncovered" and all sets in $\mathcal{I}$ "unchosen"
- Stage 1: While there exists an unchosen set $S \in \mathcal{I}$ containing at least $\frac{n}{m^\epsilon\sqrt{k}}$ uncovered items
    - Let $\widehat{S}$ be the set of uncovered items in S.

  - Relabel all items in $\widehat{S}$ as "covered" and "significant"
  - Relabel S as "chosen" and "significant"

- Stage 2: For every remaining "unchosen" set S
  - Choose $\frac{n}{m^{2\epsilon}}$ "uncovered" items $\widehat{S} \subseteq S$ uniformly at random from the uncovered items in S (if fewer than $\frac{n}{m^{2\epsilon}}$ such items, then let $\widehat{S}$ be all of them).
  - Relabel each item in $\widehat{S}$ as "covered" and "insignificant"
  - Relabel S as "chosen" and "insignificant"

- Label every uncovered item as "uncovered" and "insignificant"

When presented with a query $Q \subseteq \mathcal{X}$, we use the stored datastructure $(\mathcal{X}, \widehat{\mathcal{I}})$ in the obvious way: namely, we find $\widehat{S_1}, \ldots, \widehat{S_k} \in \widehat{\mathcal{I}}$ maximizing $|(\bigcup_{i=1}^k \widehat{S}_i) \bigcap Q|$, and return the name of the corresponding original sets $S_1, \ldots, S_k$. However, this problem cannot be solved exactly in polynomial time in general. Nevertheless, we can instead use the greedy algorithm for max-k-cover to get a constant-factor approximation [16]; this will not affect our asymptotic guarantee on the approximation ratio. The following two lemmas complete the proof that the above oracle achieves an $O(m^\epsilon/\sqrt{k})$ approximation with $\widetilde{O}(nm^{1-2\epsilon})$ space.

Lemma 4.3. *The datastructure* $(\mathcal{X}, \widehat{\mathcal{I}})$ *can be stored using* $\widetilde{O}(nm^{1-2\epsilon})$ *bits.*

The proof of the above lemma, appearing in Appendix A.1, shows that $(\mathcal{X}, \widehat{\mathcal{I}})$ can be stored as a bipartite graph with a small number of edges, by accounting for the edges created in each stage of our algorithm.

Lemma 4.4. *For every query* Q*, the oracle returns sets* $S_1, \ldots, S_k$ *such that*

$$\mathbf{E}[|(\bigcup_{i=1}^k S_i) \bigcap Q|] \geq \frac{|(\bigcup_{i=1}^k S_i^*) \bigcap Q|}{O(m^\epsilon/\sqrt{k})}$$

for any $S_1^*, \ldots, S_k^* \in \mathcal{I}$*.*

Note that $S_1, \ldots, S_k$ are random variables in the internal coin-flips of the static stage that constructs the datastructure. The expectation in the statement of the lemma is over these random coins. The proof of the lemma, appearing in Appendix A.1, distinguishes two cases depending on whether the majority of the items covered by the optimal solution appears in significant or insignificant sets.

4.4 An $O(\sqrt{n/k})$ Approximation with $\widetilde{O}(n)$ Space

This coverage oracle is similar to the one in the previous section, though is much simpler. Moreover, it is deterministic. Indeed, we construct the datastructure by the following greedy algorithm that resembles the greedy algorithm for max-k-cover

- Label all items in $\mathcal{X}$ "uncovered" and all sets in $\mathcal{I}$ "unchosen"

- While there are unchosen sets
  - Find the unchosen set $S \in \mathcal{I}$ containing the most uncovered items
  - Let $\widehat{S}$ be the set of uncovered items in S.
  - Relabel all items in $\widehat{S}$ as "covered"
  - Relabel S as "chosen"

Observe that $\widehat{\mathcal{I}}$ is a partition of $\mathcal{X}$. When presented with a query $Q \subseteq \mathcal{X}$, we use the datastructure $(\mathcal{X}, \widehat{\mathcal{I}} = \left\{\widehat{S} : S \in \mathcal{I}\right\})$ in the obvious way. Namely, we find the sets $\widehat{S_1}, \ldots, \widehat{S_k} \in \widehat{\mathcal{I}}$ maximizing $|(\bigcup_{i=1}^k \widehat{S}_i) \bigcap Q|$, and output the corresponding non-sparse sets $S_1, \ldots, S_k$. This can easily be done in polynomial time by using the obvious greedy algorithm, since $\widehat{\mathcal{I}}$ is a partition of $\mathcal{X}$.

Note that the oracle described above is very similar to the oracle from Section 4.2: The dynamic stage is identical. The static stage, however, needs to build the partition using a specific greedy ordering – as opposed to the arbitrary ordering used in Section 4.2. The following two Lemmas complete the proof that the oracle achieves an $O(\sqrt{n/k})$ approximation with $\widetilde{O}(n)$ space.

Lemma 4.5. *The datastructure* $(\mathcal{X}, \widehat{\mathcal{I}})$ *can be stored using* $\widetilde{O}(n)$ *bits*

Proof. Observe that each item is contained in exactly one $\widehat{S} \in \widehat{\mathcal{I}}$. Therefore, the bipartite graph representing the set system $(\mathcal{X}, \widehat{\mathcal{I}})$ has at most n edges. This establishes the Lemma. □

Lemma 4.6. *For every query* Q*, the oracle returns sets* $S_1, \ldots, S_k$ *with*

$$|(\bigcup_{i=1}^k S_i) \bigcap Q| \geq \frac{|(\bigcup_{i=1}^k S_i^*) \bigcap Q|}{O(\sqrt{n/k})}$$

for any $S_1^*, \ldots, S_k^* \in \mathcal{I}$*.*

The full proof of the lemma, appearing in Appendix A.1, considers two cases based on whether the majority of the elements covered by an optimal choice are in big or small sets.

5. LOWER BOUNDS

This section develops lower bounds for the SDC problem. We consider deterministic oracles that store a datastructure of size $b(n, m, k)$ for set systems with n items, m sets, maximum number of allowed sets k. Moreover, we assume that $n \leq b(n, m, k) \leq nm$, since no nontrivial positive result is possible when $b(n, m, k) = o(n)$, and a perfect approximation ratio of 1 is possible when $b(n, m, k) = \Omega(nm)$.

5.1 Main Result and Roadmap

The main result of this section is stated in the following theorem, which says that our randomized oracle in the previous section achieves a space-approximation tradeoff that essentially matches the best possible for any deterministic oracle.

Theorem 5.1. *Consider any deterministic oracle that stores a datastructure of size at most* $b(n, m, k)$ *bits, where* $n \leq b(n, m, k) \leq nm$*. Let* $\epsilon(n, m, k)$ *be such that* $b(n, m, k) =$

$nm^{1-2\epsilon(n,m,k)}$. *When $m^{\epsilon(n,m,k)} \le \sqrt{n}$, the oracle does not attain an approximation ratio of $O(\frac{m^{\epsilon(n,m,k)-\delta}}{k\sqrt{k}})$ for any constant $\delta > 0$. Moreover, when $\sqrt{n} \le m^{\epsilon(n,m,k)}$ the oracle does not attain an approximation ratio of $O(\frac{n^{1/2-\delta}}{k\sqrt{k}})$ for any $\delta > 0$.*

The proof of the theorem above is somewhat involved. Therefore, to simplify the presentation we prove in Section 5.2 a slight simplification of Theorem 5.1 that captures all the main ideas: Our simplification sets $k = 1$, and proves the $O(\frac{m^{\epsilon(n,m,k)-\delta}}{k\sqrt{k}})$ approximation ratio, for $m^{\epsilon(n,m,k)} \le \sqrt{n}$. Then, in Section 5.3 we prove the approximation ratio for the case of $\sqrt{n} \le m^{\epsilon(n,m,k)}$, still maintaining $k = 1$. Finally, in Section 5.4, we demonstrate how to modify our proofs for any k, yielding Theorem 5.1.

We fix $\delta > 0$. For the remainder of the section, we use b and ϵ as shorthand for $b(n,m,k)$ and $\epsilon(n,m,k)$, respectively. We let $\alpha(n,m,k)$ be the approximation ratio of the oracle, and use α as shorthand. Observe that $0 \le \epsilon \le 1/2$.

5.2 Proof of a Simpler Lowerbound

We simplify Theorem 5.1 by assuming $k = 1$ and $m^\epsilon \le \sqrt{n}$. The result is the following proposition, stated using the shorthand notation described above.

PROPOSITION 5.2. *Fix $k = 1$ and parameter ϵ with $0 \le \epsilon \le 1/2$. Assume $m^\epsilon \le \sqrt{n}$. Consider any deterministic oracle that stores a datastructure of size at most $b = nm^{1-2\epsilon}$ bits. The oracle does not attain an approximation ratio of $O(m^{\epsilon-\delta})$ for any constant $\delta > 0$.*

We assume the approximation ratio α attained by the oracle is $O(m^{\epsilon-\delta})$ and derive a contradiction. The proof uses the probabilistic method (see [7]). We begin by defining a distribution on set systems, and then go on to show that this distribution "fools" a small coverage oracle with positive probability.

5.2.1 Defining a Distribution D on Set Systems

We will show that there is a set system $(\mathcal{X}, \mathcal{I})$ and a query Q that forces the algorithm to output a set $S \in \mathcal{I}$ that is not within α from optimal. We use the probabilistic method. Namely, we exhibit a distribution D over set systems $(\mathcal{X}, \mathcal{I})$ such that, for every deterministic oracle storing a datastructure of size b, there exists with non-zero probability a query Q for which the oracle outputs a set of approximation worse than α. To show this, we draw two set systems i.i.d from D, and show that with non-zero probability both the following hold: the two set systems are not distinguished by the coverage oracle, and moreover there exists a query Q that requires that the algorithm return different answers for the two set systems for a $O(m^{\epsilon-\delta})$ approximation.

We define D as follows. Given the ground set $\mathcal{X} = \{1, \ldots, n\}$, we let $\mathcal{I} = \{A_i\}_{i=1}^m$ and draw $A_1, \ldots, A_m$ i.i.d as follows: We let A_i be a subset of $\mathcal{X}$ of size $nm^{-\epsilon}$ drawn uniformly at random.

5.2.2 Sampling twice from D and collisions

Next, we draw two set systems $(\mathcal{X}, \mathcal{I} = \{A_i\}_{i=1}^m)$ and $(\mathcal{X}, \mathcal{I}' = \{A'_i\}_{i=1}^m)$ i.i.d from D, as discussed above. First, we lowerbound the probability that $(\mathcal{X}, \mathcal{I})$ and $(\mathcal{X}, \mathcal{I}')$ are not distinguished by the coverage oracle. We call such an occurence a "Collision". The following result, proved in Appendix A.2, lower bounds the probability of a collision.

LEMMA 5.3. *The probability that the same datastructure is stored for $(\mathcal{X}, \mathcal{I})$ and $(\mathcal{X}, \mathcal{I}')$ is at least 2^{-b}.*

5.2.3 Fooling Queries and Candidates

Next, we lowerbound the probability that a query Q exists requiring two different answers for $(\mathcal{X}, \mathcal{I})$ and $(\mathcal{X}, \mathcal{I}')$ in order to get the desired $\alpha = O(m^{\epsilon-\delta})$ approximation. We call such a query Q a *fooling query*. We define a set of queries that are "candidates" for being a fooling query: A set $Q \subseteq \mathcal{X}$ is called a *candidate query* if $Q = A_i \bigcup A'_{i'}$ for some $i \ne i'$. In other words, a query is a candidate if it is the union of a set from $(\mathcal{X}, \mathcal{I})$ and a set from $(\mathcal{X}, \mathcal{I}')$ with different indices.

Ideally, candidate $Q = A_i \bigcup A'_{i'}$ would be a fooling query by forcing the oracle to output i for $(\mathcal{X}, \mathcal{I})$ and i' for $(\mathcal{X}, \mathcal{I}')$ in order to guarantee the desired approximation. However, this need not be the case: consider for instance the case when, for some $j \ne i, i'$, both A_j and A'_j have large intersection with Q, making it ok to output j for both. We will show that the probability that none of the candidate queries is a fooling query is strictly less than 2^{-b} when n and m are sufficiently large. Doing so would complete the proof: collision occurs with probability $\ge 2^{-b}$, and a fooling query exists with probability $> 1 - 2^{-b}$, and therefore both occur simultaneously with positive probability. This would yield the desired contradiction.

5.2.4 The Probability that None of the Candidates is Fooling is Small

We now upperbound the probability that none of the candidates is a fooling query. Observe that if candidate $Q = A_i \bigcup A'_{i'}$ is not a fooling query, then there exists $A \in \mathcal{I} \bigcup \mathcal{I}' \setminus \{A_i, A'_{i'}\}$ with $|A \bigcap Q| \ge nm^{-\epsilon}/\alpha$. Therefore one of the following must be true:

1. There exists $A \in \mathcal{I} \bigcup \mathcal{I}' \setminus \{A_i, A'_{i'}\}$ with $|A \bigcap A_i| \ge nm^{-\epsilon}/2\alpha = \Omega(nm^{-2\epsilon+\delta})$.
2. There exists $A \in \mathcal{I} \bigcup \mathcal{I}' \setminus \{A_i, A'_{i'}\}$ with $|A \bigcap A'_{i'}| \ge nm^{-\epsilon}/2\alpha = \Omega(nm^{-2\epsilon+\delta})$.

Therefore, if none of the candidates were fooling queries, then there are many "pairs" of sets in $\mathcal{I} \bigcup \mathcal{I}'$ that have an intersection substantially larger than the expected size of $nm^{-2\epsilon}$. This seems very unlikely. Indeed, the remainder of this proof will demonstrate just that.

If none of the candidates are fooling queries, then by examining (1) and (2) above we deduce the following. There exists [3] a set of pairs $P \subseteq (\mathcal{I} \bigcup \mathcal{I}') \times (\mathcal{I} \bigcup \mathcal{I}')$ such that:

1. $|P| \ge m - 2 = \Omega(m)$
2. The undirected graph with nodes $\mathcal{I} \bigcup \mathcal{I}'$ and edges P is bipartite. Moreover, every node in the left part has degree at most 1. Thus P is acyclic.

[3] Consider constructing P as follows: For candidate query $Q = A_1 \bigcup A'_2$, find the set in $\mathcal{I} \bigcup \mathcal{I}' \setminus \{A_1, A'_2\}$ with a large intersection with one of A_1 or A'_2 as in (1) or (2). Say for instance we find that A_7 has a large intersection with A_1. We include (A_1, A_7) in P, mark both A_1 and A_7 as "touched", and designate A_1 a "left" node and A_7 a "right" node. Then, we repeat the process with some candidate $Q' = A_i \bigcup A'_{i'}$ for some "untouched" A_i and $A'_{i'}$. We keep repeating until there are no such candidates. Throughout this greedy process, we mark at most two members of $\mathcal{I} \bigcup \mathcal{I}'$ as "touched" for every pair we include in P. Note that some A_i may be "touched" more than once. As long as there are at least 2 untouched sets in each of $\mathcal{I}$ and $\mathcal{I}'$, the algorithm may continue.

3. If $(B, C) \in P$ then $|B \bigcap C| \geq \Omega(nm^{-2\epsilon+\delta})$

We now proceed to bound the probability of existence of such a P, and in the process also bound the probability that none of the candidate queries are fooling. Recall that members of $\mathcal{I} \bigcup \mathcal{I}'$ are drawn i.i.d from the uniform distribution on subsets of $\mathcal{X}$ of size $nm^{-\epsilon}$. For every pair $(B, C) \in \mathcal{I} \bigcup \mathcal{I}'$, we let $\mathcal{R}(B, C) = |B \bigcap C|$ denote the size of their intersection. It is easy to see the random variables $\{\mathcal{R}(B, C)\}_{B,C \in \mathcal{I} \bigcup \mathcal{I}'}$ are pairwise independent. Therefore, any acyclic set of pairs is mutually independent, by basic probability theory. Thus, if we fix a particular P satisfying (1) and (2), the probability that P satisfies condition (3) is at most

$$\prod_{(B,C) \in P} \mathbf{Pr}[\mathcal{R}(B, C) \geq \Omega(nm^{-2\epsilon+\delta})]$$

We now want to estimate the probability that the intersection of B and C is a factor $\Omega(m^\delta)$ more than its expectation of $nm^{-2\epsilon}$. Therefore, we consider an indicator random variable Y_i for each $i \in \mathcal{X}$, designating wheter $i \in B \cap C$. If Y_i were independent, we could use Chernoff bounds to bound the probability that $\mathcal{R}(B, C)$ is large. Fortunately, it is easy to see that the Y_i's are negatively-correlated: i.e., for any $L \subseteq \{1, \ldots, n\}$, we have $\mathbf{Pr}[\bigwedge_{i \in L} Y_i = 1] \leq \prod_{i \in L} \mathbf{Pr}[Y_i = 1]$. Therefore, by the result of [17], if we "pretend" that they are independent by approximating their joint-distribution by i.i.d bernoulli random variables, we can still use Chernoff Bounds to bound the upper-tail probability. Therefore, using Chernoff bounds[4] we deduce that the probability that the intersection of B and C is a factor $\Omega(m^\delta)$ more than the expectation of $nm^{-2\epsilon}$ is at most $2^{-(\Omega(m^\delta)-1)nm^{-2\epsilon}} \leq 2^{-\Omega(nm^{-2\epsilon+\delta})}$. Therefore, the probability that the fixed P satisfies condition (3) is at most

$$\prod_{(B,C) \in P} 2^{-\Omega(nm^{-2\epsilon+\delta})} \leq (2^{-\Omega(nm^{-2\epsilon+\delta})})^{|P|} \leq 2^{-\Omega(nm^{1-2\epsilon+\delta})}$$

Now, we can sum over all possible choices for P satisfying (1) and (2) to get a bound on the existence of a P satisfying (1), (2) and (3). It is easy to see that there are at most m^m choices for P that satify (1) and (2). Using the union bound, we get the following bound on the existence of such a P.

$$m^m \cdot 2^{-\Omega(nm^{1-2\epsilon+\delta})} \leq 2^{m \log m - \Omega(nm^{1-2\epsilon+\delta})} \leq 2^{-\Omega(nm^{1-2\epsilon+\delta})}$$

Where the last inequality follows by simple algebraic manipulation from our assumption that $m^\epsilon \leq \sqrt{n}$ and $\delta > 0$, when n and m are sufficiently large. Recall that, by our previous discussion, this expression also upperbounds the probability that none of the candidate queries are fooling queries. But, when n and m are sufficiently large, this is strictly smaller than $2^{-b} = 2^{-nm^{1-2\epsilon}}$. Thus, by our previous discussion, this completes the proof of Proposition 5.2.

5.3 Modifying the proof for the case $\sqrt{n} \leq m^\epsilon$

We maintain the assumption that $k = 1$, and extend Proposition 5.2 for the case when $\sqrt{n} \leq m^\epsilon$; the proof of the following result is similar to that of Proposition 5.2, and appears in Appendix A.2.

PROPOSITION 5.4. *Fix $k = 1$ and parameter ϵ with $0 \leq \epsilon \leq 1/2$. Assume $\sqrt{n} \leq m^\epsilon$. Consider any deterministic oracle that stores a datastructure of size at most $b = nm^{1-2\epsilon}$ bits. The oracle does not attain an approximation ratio of $O(n^{1/2-\delta})$ for any constant $\delta > 0$.*

5.4 Modifying the proof for arbitrary k

In this section, we generalize Proposition 5.2 to arbitrary k. The generalization of Proposition 5.4 to arbitrary k is essentially identical, and therefore we leave it as an exercise for the reader. We now state the generalization of Proposition 5.2 to arbitrary k, the proof of which again follows Proposition 5.2 and appears in Appendix A.2.

PROPOSITION 5.5. *Let parameter ϵ be such that $0 \leq \epsilon \leq 1/2$. Assume $m^\epsilon \leq \sqrt{n}$. Consider any deterministic oracle that stores a datastructure of size at most $b = nm^{1-2\epsilon}$ bits. The oracle does not attain an approximation ratio of $O(\frac{m^{\epsilon-\delta}}{k\sqrt{k}})$ for any constant $\delta > 0$.*

6. CONCLUSIONS AND FUTURE WORK

This paper introduced and studied a fundamental problem, called SDC, arising in many large-scale Web applications. A summary of results obtained by the paper appear in Table 1 (Section 3.2). The main specific open question that arises is whether there is a deterministic oracle that is as good as the randomized oracle proposed in Section 4. More generally, a detailed analysis of practical subclasses of SDC seems to hold promise.

7. REFERENCES

[1] Amazon. *www.amazon.com.*

[2] eBay. *www.ebay.com.*

[3] Netflix. *http://www.netflix.com.*

[4] Netflix Prize. *http://www.netflixprize.com.*

[5] Yelp. *www.yelp.com.*

[6] Gediminas Adomavicius and Er Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE TKDE*, 17, 2005.

[7] N. Alon and J. Spencer. *The Probabilistic Method.* John Wiley, 1992.

[8] Y. Azar ÁÍÄ N. Alon ÁÍÄ B. Awerbuch. The online set cover problem. In *STOC*, 2003.

[9] Francesco Bonchi, Carlos Castillo, Debora Donato, and Aristides Gionis. Topical query decomposition. In *KDD*, 2008.

[10] Jaime G. Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, 1998.

[11] M. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, 2002.

[12] Harr Chen and David R. Karger. Less is more: probabilistic models for retrieving fewer relevant documents. In *SIGIR*, 2006.

[13] Nello Cristianini and Matthew W. Hahn. *Introduction to Computational Genomics.* Cambridge Unversity Press, 2006.

[4] We use the following version of the Chernoff Bound: Let $X_1, \ldots, X_n$ be independent bernoulli random variables, and let $X = \sum_i X_i$. If $\mathbf{E}[X] = \mu$ and $\Delta > 2e - 1$, then $\mathbf{Pr}[X > (1 + \Delta)\mu] \leq 2^{-\Delta\mu}$.

[14] M. R. Garey and D. S. Johnson. Computers and Intractability. *W. H. Freeman and Company*, 1979.

[15] YJ. Naor and N. Buchbinder. Online primal-dual algorithms for covering and packing problems. In *ESA*, 2005.

[16] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions – I. *Mathematical Programming*, 14(3):265–294, 1978.

[17] Alessandro Panconesi and Aravind Srinivasan. Randomized distributed edge coloring via an extension of the chernoff-hoeffding bounds. *SIAM J. Comput.*, 26(2):350–368, 1997.

[18] B. Saha and L. Getoor. On maximum coverage in the streaming model and application to multi-topic blog-watch. In *SDM*, 2009.

[19] Mikkel Thorup and Uri Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24, 2005.

APPENDIX

A. PROOFS

A.1 Upper Bound Proofs

Proof of Lemma 4.3: We store the set system as a bipartite graph representing the containment relation between items and sets. To show that the bipartite graph can be stored in the required space, it suffices to show that $(\mathcal{X}, \widehat{\mathcal{I}})$ is "sparse"; namely, that the total number of edges $(x, \widehat{S}) \in \mathcal{X} \times \widehat{\mathcal{I}}$ such that $x \in \widehat{S}$ is $O(nm^{1-2\epsilon})$. We account for the edges created in stages 1 and 2 separately.

1. Every significant item is connected to a single set. This creates at most n edges.
2. For every insignificant set, we store $\leq nm^{-2\epsilon}$ items, creating at most $mnm^{-2\epsilon} = nm^{1-2\epsilon}$ edges.

□

Proof of Lemma 4.4: We fix an optimal choice for $S_1^*, \dots, S_k^* \in \mathcal{I}$, and denote $OPT = |(\bigcup_{i=1}^k S_i^*) \bigcap Q|$. Since, by construction, $\widehat{S} \subseteq S$ for all $S \in \mathcal{I}$, it suffices to show that the output of the oracle satisfies $|(\bigcup_{i=1}^k \widehat{S}_i) \bigcap Q| \geq \frac{OPT}{O(m^\epsilon/\sqrt{k})}$ in expectation. Moreover, since the dynamic stage algorithm finds a constant factor approximation to $\max\{|(\bigcup_{i=1}^k \widehat{S}_i) \bigcap Q| : \widehat{S_1}, \dots, \widehat{S_k} \in \widehat{\mathcal{I}}\}$, it is sufficient to show that there *exists* $S_1, \dots, S_k \in \mathcal{I}$ with $\mathbf{E}[|(\bigcup_{i=1}^k \widehat{S}_i) \bigcap Q|] \geq \frac{OPT}{O(m^\epsilon/\sqrt{k})}$.

We distinguish two cases, based on whether most of the items $(\bigcup_{i=1}^k S_i^*) \bigcap Q$ covered by the optimal solution are in significant or insignificant sets. We use the "significant" and "insignificant" designation as used in the static stage algorithm. Moreover, we refer to $\widehat{S} \in \widehat{\mathcal{I}}$ as significant (insignificant, resp.) when the corresponding $S \in \mathcal{I}$ is significant (insignificant, resp.).

1. **At least half of $(\bigcup_{i=1}^k S_i^*) \bigcap Q$ are significant items**: Notice that, by construction, there are at most $m^\epsilon\sqrt{k}$ significant sets in $\widehat{\mathcal{I}}$. Moreover, the significant items are precisely those covered by the significant sets of $\widehat{\mathcal{I}}$, and those sets form a partition of the significant items. Therefore, by the pigeonhole principle, there are there are some $\widehat{S_1}, \dots, \widehat{S_k} \in \widehat{\mathcal{I}}$ such that $\bigcup_{i=1}^k \widehat{S}_i$ contains at least an $\frac{k}{m^\epsilon\sqrt{k}} = \frac{\sqrt{k}}{m^\epsilon}$ fraction of the significant items in $(\bigcup_{i=1}^k S_i^*) \bigcap Q$. This gives the desired $O(m^\epsilon/\sqrt{k})$ approximation.
2. **At least half of $(\bigcup_{i=1}^k S_i^*) \bigcap Q$ are insignificant items**: In this case, at least half the items $(\bigcup_{i=1}^k S_i^*) \bigcap Q$ covered by the optimal solution are contained in the insignificant members of $\{S_1^*, \dots, S_k^*\}$. Recall that any insignificant set in $\mathcal{I}$ contains at most $\frac{n}{m^\epsilon\sqrt{k}}$ insignificant items. Therefore, the algorithm includes each element of an insignificant S_i^* in $\widehat{S_i^*}$ with probability at least $\frac{n}{m^{2\epsilon}} / \frac{n}{m^\epsilon\sqrt{k}}$, which is at least $\sqrt{k}/m^\epsilon$. Thus, every insignificant item in $(\bigcup_{i=1}^k S_i^*)$ is in $(\bigcup_{i=1}^k \widehat{S_i^*})$ with probability at least $\sqrt{k}/m^\epsilon$. This gives that the expected size of $(\bigcup_{i=1}^k \widehat{S_i^*}) \bigcap Q$ is at least $\frac{OPT}{O(m^\epsilon/\sqrt{k})}$. Taking $S_i = S_i^*$ completes the proof.

□

Proof of Lemma 4.6: Fix an optimal choice of $S_1^*, \dots, S_k^*$, and denote $OPT = |(\bigcup_{i=1}^k S_i^*) \bigcap Q|$. Recall that the oracle finds $\widehat{S_1}, \dots, \widehat{S_k} \in \widehat{\mathcal{I}}$ maximizing $|(\bigcup_{i=1}^k \widehat{S}_i) \bigcap Q|$, and then outputs the corresponding original sets $S_1, \dots, S_k$.

It suffices to show that there are some $\widehat{S}_1, \dots, \widehat{S}_k \in \widehat{\mathcal{I}}$ with $|(\bigcup_{i=1}^k \widehat{S}_i) \bigcap Q| \geq OPT/O(\sqrt{n/k})$. We distinguish two cases, based on whether most of $(\bigcup_{i=1}^k S_i^*) \bigcap Q$ are in big or small sets in $\widehat{\mathcal{I}}$.

Recall that $\widehat{\mathcal{I}}$ forms a partition of $\mathcal{X}$. We say $\widehat{S} \in \widehat{\mathcal{I}}$ is "significant" if $|\widehat{S}| \geq \sqrt{n/k}$, otherwise $\widehat{S}$ is "insignificant". Similarly, we say an item $i \in \mathcal{X}$ is "significant" if it falls in a significant set in $\widehat{\mathcal{I}}$, otherwise it is "insignificant". Notice that there are at most $\frac{n}{\sqrt{n/k}} = \sqrt{nk}$ significant sets.

First, we consider the case where at least half the items in $(\bigcup_{i=1}^k) S_i^* \bigcap Q$ are significant. Since there at most $\sqrt{nk}$ significant sets in $\widehat{\mathcal{I}}$, by the pigeonhole principle there are k of them that collectively cover a $k/\sqrt{nk} = \sqrt{k/n}$ fraction of all significant items in $(\bigcup_{i=1}^k S_i^*) \bigcap Q$. This would guarantee the $O(\sqrt{n/k})$ approximation, as needed.

Next, we consider the case where at least half of $(\bigcup_{i=1}^k S_i^*) \bigcap Q$ are insignificant. By examining the greedy algorithm of the static stage, it is easy to see that each $S \in \mathcal{I}$ contains at most $\sqrt{n/k}$ insignificant items. Therefore, there are at most $k \cdot \sqrt{n/k} = \sqrt{nk}$ insignificant items in $(\bigcup_{i=1}^k S_i^*)$. Therefore we deduce that $OPT = |(\bigcup_i S_i^*) \bigcap Q| \leq 2\sqrt{kn}$. Since the optimal covers $O(\sqrt{kn})$ items in Q, it suffices for a $O(\sqrt{n/k})$ approximation to show that there are $\widehat{S}_1, \dots, \widehat{S}_k \in \widehat{\mathcal{I}}$ that collectively cover k items of Q. It is easy to see that this is indeed the case, since $\widehat{\mathcal{I}}$ is a partition of $\mathcal{X}$. This completes the proof. □

A.2 Lower Bound Proofs

A.2.1 Proof of Lemma 5.3

There are 2^b possible datastructures. Let p_i denote the probability that, when presented with random $(\mathcal{X}, \mathcal{I}) \sim D$, the oracle stores the i'th datastructure. We can write this probability of "collision" of the two i.i.d samples $(\mathcal{X}, \mathcal{I})$ and $(\mathcal{X}, \mathcal{I}')$ as $\sum_{i=1}^{2^b} p_i^2$. However, since $\sum_i p_i = 1$, this expression is minimized when $p_i = 2^{-b}$ for all i. Plugging into the above expression gives a lowerbound of 2^{-b}, as required. □

A.2.2 Proof of Proposition 5.4

Instead of replicating almost the entire proof of Proposition 5.2, we instead point out the key changes necessary to yield a proof of 5.4 and leave the rest as an easy excercise for the reader.

The proof proceeds almost identically to the proof of Proposition 5.2, with the following main changes:

- **Modifications to Section 5.2.1**: When defining D, we let each A_i be a subset of $\mathcal{X}$ of size $\sqrt{n}$ instead of $nm^{-\epsilon}$.
- We perform similar calculations throughout, accomodating the above modification to the size of A_i.
- **Modifications to Section 5.2.4**: We eventually arrive at an upper bound of $2^{-mn^{\delta}}$ on the probability that none of the candidate queries are fooling. Using the assumption $m^{\epsilon} \geq \sqrt{n}$ and the fact that $b = nm^{1-2\epsilon}$, a simple algebraic manipulation shows that this bound is stricly less than 2^{-b}. This completes the proof, as before.

A.2.3 Proof of Proposition 5.5

The proof of Proposition 5.5 follows the outline of the proof of Proposition 5.2. The necessary modifications to the proof of Proposition 5.2 are as follows:

- **Modifications to Section 5.2.1**: We define distribution D as before, except that we let each A_i be a subset of $\mathcal{X}$ of size $nm^{-\epsilon}\sqrt{k}$.
- **Modifications to Section 5.2.2**: Instead of sampling from D twice, we sample $2k+1$ times to get set systems $(\mathcal{X}, \mathcal{I}^1), (\mathcal{X}, \mathcal{I}^2), \ldots, (\mathcal{X}, \mathcal{I}^{2k+1})$. This changes the probability of collision of Lemma 5.3 to 2^{-2kb}. Here, collision means that all $2k{+}1$ samples from D are stored as the same datastructure by the static stage of the oracle.
- **Modifications to Section 5.2.3**: We now define a *fooling query* analogously for general k: A query Q is fooling if there is no single index i such that returning the i'th set gives a good approximation for all the set systems $(\mathcal{X}, \mathcal{I}^1), \ldots (\mathcal{X}, \mathcal{I}^{2k+1})$.

  Moreover, we analogously define *candidate queries*: We use A_b^a to denote the b'th set in set system $(\mathcal{X}, \mathcal{I}^a)$. We say $Q \subseteq \mathcal{X}$ is a candidate if $Q = A_{i_1}^{\ell_1} \bigcup A_{i_2}^{\ell_2} \bigcup \ldots \bigcup A_{i_{k+1}}^{\ell_{k+1}}$, where indices $\ell_1, \ldots, \ell_{k+1}$ are distinct, and indices $i_1, \ldots, i_{k+1}$ are distinct. In other words, Q is a fooling query if it is the union of $k+1$ sets from $k+1$ distinct set systems and $k+1$ distinct indices in those set systems.
- **Modifications to Section 5.2.4**: Similarly, if a candidate $Q = A_{i_1}^{\ell_1} \bigcup \ldots \bigcup A_{i_{k+1}}^{\ell_{k+1}}$ is not a fooling query, then there is some $A \in (\mathcal{I}^1 \bigcup \ldots \mathcal{I}^{2k+1}) \setminus \left\{ A_{i_j}^{\ell_j} \right\}_j$ with $|A \bigcap Q| \geq nm^{-\epsilon}/\alpha$. Therefore, for one of the components $A_{i_j}^{\ell_j}$ of Q we have that $|A \bigcap A_{i_j}^{\ell_j}| \geq nm^{-\epsilon}/k\alpha$. Plugging in the approximation ratio $\alpha = m^{\epsilon-\delta}/k\sqrt{k}$ we have that $|A \bigcap A_{i_j}^{\ell_j}| \geq nm^{-2\epsilon+\delta}\sqrt{k}$. It is not too hard to see that we can construct P similarly with

  1. $|P| \geq k(m-k) = \Omega(km)$.[5]
  2. The undirected graph with nodes $\mathcal{I} \bigcup \mathcal{I}'$ and edges P is bipartite. Moreover, every node in the left part has degree at most 1. Thus P is acyclic.
  3. If $(B, C) \in P$ then $|B \bigcap C| \geq \Omega(nm^{-2\epsilon+\delta}\sqrt{k})$

  Continuing with the remaining calculations in this section almost identically gives a bound of $2^{-\Omega(knm^{1-2\epsilon+\delta})}$ on the probability of existance of a fixed P. The number of such P is at most $(km)^{km}$, therefore a similar calculation gives a bound of

  $$2^{-\Omega(knm^{1-2\epsilon+\delta})} = 2^{-\Omega(kbm^{\delta})}$$

  on the *existence* of any such P. As before, this completes the proof.

Acknowledgements

We thank Philip Bohannon, Hector Garcia-Molina, Ashwin Machavanajhhala, Tim Roughgarden, and Elad Verbin for insightful discussions.

[5] This is not true when k is almost equal to m. However, the theorem becomes trivially true when $k > m^{1/6}$, so we can without loss assume that k is not too large.