Finding Witnesses by Peeling

Yonatan Aumann* Moshe Lewenstein* Noa Lewenstein† Dekel Tsur‡

Abstract

In the k-matches problem, we are given a pattern and a text, and for each text location, the desired output consists of all aligned matching characters if there are k or fewer of them, and any k aligned matching characters if there are more than k of them. This problem is one of several string matching problems that seek not only to find where the pattern matches the text, under different "match" definitions, but also to provide *witnesses* to the match. Other such problems include: k-aligned ones [5], k-witnesses, and k-mismatches [23]. In addition, the solutions to several other string matching problems rely on the efficient solutions of the witness finding problems.

In this paper we provide a general method for solving such witness finding problems efficiently. We do so by casting the problem as a generalization of group testing, which we then solve by a process we call *peeling*. Using this general framework we obtain improved results for all of the above problems. We also show that our method also solves a couple of problems outside the pattern matching domain.

1 Introduction

Pattern matching is a well studied domain with a large collection of problems. In classical pattern matching the goal is to find all occurrences of a pattern in a text [19]. One avenue of research within pattern matching is the extension of this problem to allow for more general matching criteria. Another is to allow a bounded number of errors to occur. In both of these extensions the desired output is the locations that match, or the locations which mismatch, under the extended criteria or other limitations.

In several applications it is not sufficient to output the locations which matched or mismatched. Rather we will want a *witness* to the match or mismatch. For example, consider the classical pattern matching problem. If we have a pattern $p =$ 'aba' and a text $t =$ 'aabaa'. Then the pattern matches at text location 2 but not at any other text location. The $|p|$-length substring beginning at text location 1 is 'aab'. A witness to the mismatch at location 1 is the

*Dept. of Computer Science, Bar-Ilan University, Ramat Gan, Israel.

†School of Computer Science, Netanya Academic College, Netanya, Israel.

‡Dept. of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel. The research was supported by the Lynn and William Frankel Center for Computer Sciences at Ben-Gurion University.

second character of the pattern as it is ‘b’ in p and ‘a’ in ‘aab’. Of course, match is a general term which is application dependent and also witness needs change with the application. Finding witnesses arises in several applications which we mention below. However, doing so efficiently is not a simple task.

The concept of finding a “witness” was independently introduced for applications of *boolean matrix multiplication*, specifically for the all-pairs shortest path. Formally let C be the (boolean) multiplication of A and B. Then if $c_{i,j} = 1$, we say that t is a *witness* if $A_{i,t} = 1$ and $B_{t,j} = 1$.

We now mention several applications where finding k witnesses is desired.

Aligned Ones and Matches Let $t = t_0 \cdots t_{n-1}$ be a text and $p = p_0 \cdots p_{m-1}$ be a pattern. Consider an alignment of p with t starting at location i in t, namely p_0 is aligned with t_j, p_1 is aligned with t_{j+1}, and so on. We say that $(j, i + j)$ is a *match pair* in the alignment if $p_j = t_{i+j}$, and otherwise $(j, i + j)$ is a mismatch pair in the alignment. The *k-aligned ones problem*, defined in [5] and further explored in [23], is the following. For text and pattern both over the alphabet $\Sigma = \{0, 1\}$, for every alignment of p with t which has at most k match pairs of the character 1, output all of these match pairs (in this problem, we ignore match pairs of the character 0). Note that no output is required for alignments that have more than k match pairs. The k-aligned ones problem was used in [5] to solve a two-dimensional pattern matching problem called half-rectangular matching.

In the *k-witnesses problem*, the goal is to find k match pairs of the character 1 for every alignment of p with t (for alignments with fewer than k match pairs of 1, all of the match pairs of 1 in these alignments need to be found). The solution in [5] to the k-aligned ones problem is a very interesting, but rather involved and costly procedure. In [23] a simpler solution was suggested and it hints at the direction we take.

In the *k-matches problem*, the desired output consists of all the match pairs if there are k or fewer of them, and any k match pairs if there are more than k of them. In contrast to the k-witnesses problem, here all match pairs are considered and not only match pairs of the character 1. Moreover, the problem is over general alphabets whereas the k-witnesses problem is over the alphabet $\{0, 1\}$.

Mismatches under General Matching Relations A similar problem to the above is the k-mismatches problem. Here the problem is to find all alignments in which the number of mismatch pairs does not exceed k.

This problem is well-studied (e.g. [1, 6, 15, 21]), and the most efficient algorithm [6] runs in time $O(n\sqrt{k \log k})$. The algorithms of [15, 21], which run in time $O(nk)$, also enable the mismatch pairs to be reported.

All of the above works assume that the matching relation (between pattern elements and text elements) is the equality relation. In some applications, however, other “matching” relations may arise, for example wildcard matching [10, 14], smaller matching [5], and distance matching [12]. In such problems, the matching is defined by a $\Sigma \times \Sigma$ matching matrix M, where Σ is the alphabet. The *k-mismatches problem* [23] is the problem of finding, for each alignment of p with t, at least k mismatch pairs (under M) if possible; if there are fewer than k mismatch pairs, of finding all mismatch pairs.

For general matching relations M the suffix tree methods of [15, 21] do not work. Muthukrishnan [23, Theorem 5] showed a relationship between the k-mismatches problem and the k-witnesses problem.

Matrix Multiplication Witnesses and All-Pairs k-Shortest Path Let A and B be two boolean $n \times n$ matrices, and let $C = A \cdot B$, where multiplication is the logical $\wedge$ and addition the logical $\vee$. The *witness for boolean matrix multiplication problem* is to find, for each i, j such that $c_{i,j} = 1$, an index t such that $a_{i,t} = b_{t,j} = 1$. Alon, Galil, Margalit, and Naor [3] and Seidel [24] independently discovered a simple randomized algorithm to solve the problem. Both use the witness for boolean matrix multiplication problem to solve the all-pairs shortest path problem on undirected unweighted graphs.

A natural extension to the all-pairs shortest path problem is the problem of the *all-pairs k-shortest paths*; that is, for every pair of vertices u, v, to output the k (not necessarily disjoint) shortest paths between u and v (if they exist). It turns out that one way to solve this problem is to extend the witness problem to the *k-witnesses for boolean matrix multiplication problem*: Let $C = A \cdot B$ with multiplication over the integers. reals. For each i, j, find $k' = \min(k, c_{i,j})$ "witnesses" for $c_{i,j}$, i.e. indexes $t_1, \ldots, t_{k'}$, such that for each $t \in \{t_1, \ldots, t_{k'}\}$, $a_{i,t} = b_{t,j} = 1$. We will show in the sequel that the all-pairs k-shortest paths problem can be solved in the same time complexity as the k-witnesses for boolean matrix multiplication problem. Note that one cannot solve the latter problem by simply reiterating the procedure of the one witness problem as it may rediscover the same witness repeatedly.

1.1 The Reconstruction Problem

To solve the above-mentioned problems, we define a generalization of group testing which is a process in which we "peel witnesses", called the *reconstruction problem.* We solve several variants of the reconstruction problem in this paper and utilize them to solve our witness problems. We believe that the peeling process is interesting in its own right, although we view the main contribution of the paper as a generalized setting and solution for the set of problems mentioned below. We first describe a group testing setting and then return to show how this helps to solve the above-mentioned problems.

Let U be a universe of m distinct items, all taken from some commutative group G. The *group testing problem* is to determine the members of a set $S \subseteq U$, $|S| \leq k$, by queries of the form "does A contain an element of S?", where A is some subset of U. Our goal in group testing is to determine S with a minimal number of queries.

Many variants of group testing have been considered. One of these variants is a change of query asking for $|S \cap A|$ rather than whether $S \cap A$ is empty or not. This problem is known as *quantitative group testing*, see [13], or *interpolation set* [17, 18]. In *additive group testing*, a query asks for the value of $\sum_{u \in S \cap A} u$ for some set A.

Here we consider a generalization of the above. Consider a collection of unknown sets $S_1, \ldots, S_n \subseteq U$, and suppose that we are interested in reconstructing $S_1, \ldots, S_n$, or in finding

k elements of each S_i (for some parameter k). We are provided with procedures such that for any set $A \subseteq U$, we can compute:

- Intersection Cardinality: $\text{ISIZE}_{S_1,\dots,S_n}(A) = \langle |S_1 \cap A|, \dots, |S_n \cap A| \rangle$.
- Intersection Sum: $\text{ISUM}_{S_1,\dots,S_n}(A) = \langle \sum_{u \in S_1 \cap A} u, \dots, \sum_{u \in S_n \cap A} u \rangle$.

Clearly, given a sufficient number of calls to these procedures, it is possible to fully reconstruct $S_1, \dots, S_n$. However, we aim at doing so with a minimal amount of work. We note that an efficient algorithm for this problem must provide:

1. The list or sequence of sets A on which to compute $\text{ISIZE}_{S_1,\dots,S_n}(A)$ and/or $\text{ISUM}_{S_1,\dots,S_n}(A)$ (note that for determining the A's we do not have explicit knowledge of $S_1, \dots, S_n$),

2. An efficient procedure to reconstruct k elements of each S_i based on the information provided by these computations.

In this paper we consider two variants of the general setting outlined above. In the first variant, the sizes of $S_1, \dots, S_n$ can be arbitrary, and we are interested in finding $\min(k, |S_i|)$ elements of S_i for all i. We call this the *k-reconstruction problem.* In the second variant, we are interested in fully reconstructing every set S_i of size at most k. We call this the *bounded k-reconstruction problem.* We present efficient solutions to both these problems. The solutions are based upon a combinatorial structure, which we call *peeling collections*, and a matching algorithmic procedure, which we call the *peeling procedure.* The basic idea is as follows. If a set $A \subseteq U$ satisfies $|S_i \cap A| = 1$, then we can "peel" the unique element of the intersection of S_i by querying $\sum_{u \in S_i \cap A} u$. A *peeling collection* is a collection of sets such that for any S its elements can be peeled one by one by peeling and updating the other sets to reflect that an element has been peeled. The full details appear in Section 2.

We point out the following differences from previous works:

1. Most group testing codes are universal in the sense that the elements of any set S can be recovered using the codes. In [9] a lower bound of $\Omega(k \log m)$ was given on the construction of k-selective codes equivalent to the deterministic bounded-reconstruction (see Section 2). Here we consider reconstructing a single set S (which one can view as a special code for one unknown S) with high probability, beating the lower bounds for the k-selective codes.

2. The problem is an n-set problem as opposed to the one-set problems in the previous variants. While the problem can be solved by finding separate tests for each set, sometimes it can be solved more efficiently directly. Moreover, we show applications that fall nicely into this setting.

3. The peeling method. This gives an adaptive flavor to a non-adaptive setting. It should be mentioned that the peeling method appears implicitly in [18] yet is rather inefficient and under-exploited.

As far as we know, k-reconstruction problems have not been studied explicitly before. However, in previous works, these problems have been solved implicitly: Amir et al. [5] implicitly gave a deterministic algorithm that solves the bounded k-reconstruction problem. A deterministic algorithm for this problem can also be obtained from [18] with some simple observations. For the unbounded problem, Seidel [24] and Alon and Naor [4] implicitly solved the unbounded 1-reconstruction problem. Muthukrishnan [23] implicitly gave a randomized algorithm for solving the unbounded k-reconstruction problem.

New results We now describe our results for the reconstruction problem, which improve the previous (implicit) results. Suppose that computing $\text{ISize}_{S_1,\ldots,S_n}(A)$ and $\text{ISum}_{S_1,\ldots,S_n}(A)$ for a single set A takes $O(f)$ steps. We assume that $f = \Omega(m+n)$ as the size of the input to $\text{ISize}_{S_1,\ldots,S_n}(\cdot)$ and $\text{ISum}_{S_1,\ldots,S_n}(\cdot)$ is $\Theta(m)$ (in the worst case) and the size of the output is $\Theta(n)$. For the bounded k-reconstruction problem we present a deterministic algorithm that runs in $O(fk \cdot \text{polylog}(m))$ steps, and a randomized algorithm that runs in expected $O(f(k + \log k \cdot \log n) + nk\log(mn))$ steps. For the (unbounded) k-reconstruction problem, we give a deterministic algorithm with time complexity $O(fk\log n \cdot \text{polylog}(m))$, and a randomized algorithm with expected time complexity $O(f(k(\log m + \log k \cdot \log\log m) + \log n \cdot \log m) + nk\log n)$.

Given the efficient solutions to the general problems, we show problems that can be cast as special cases of this general framework. Accordingly, we provide improved algorithms for these problems. We now list the new results. In the following, let $T_b(n, m, k, f)$ (resp., $T_u(n, m, k, f)$) be the time needed to solve the bounded (resp., unbounded) k-reconstruction problem with parameters n, m, k, and f.

1.2 Our Results

Aligned Ones and Matches For the k-aligned ones problem, Amir and Farach [5] gave an $O(nk^3 \log m \log k)$ time deterministic algorithm, and Muthukrishnan [23] provided an $O(nk^2 \log^4 m)$ randomized algorithm. Muthukrishnan [22] improved this to an $O(nk \log^4 m)$ randomized algorithm.

We show that the k-aligned ones problem can be solved in time $O(\frac{n}{m} T_b(2m, m, k, m\log m))$. Therefore, we obtain an $O(nk \cdot \text{polylog}(m))$ deterministic algorithm, and an $O(n \log m(k + \log k \cdot \log m))$ randomized algorithm for the problem. Moreover, the k-matches problem can be solved in $O(|\Sigma| \frac{n}{m} T_u(2m, m, k, m\log m))$ time.

Mismatches under General Matching Relations Muthukrishnan [23, Theorem 5] showed that the k-mismatches problem over arbitrary M can be solved with $\text{cpn}(G)$ calls to the k-witnesses problem, where G is the conflict graph (the bipartite graph corresponding to the complement of M) and $\text{cpn}(G)$ is the minimum number of edge-disjoint bipartite cliques of G. Additionally, [23] provided an $O(n\sqrt{km \log m})$ deterministic algorithm, and an expected $O(nk^2 \log^3 m \log n)$ randomized algorithm to the k-witnesses problem. An open problem presented in [23] was to find a deterministic algorithm for the k-witnesses problem with time comparable to the randomized algorithm.

The k-witnesses problem can be solved in $O(\frac{n}{m}T_u(2m, m, k, m\log m))$ time. Therefore, we obtain an $O(\text{cpn}(G) \cdot nk \cdot \text{polylog}(m))$ time deterministic algorithm and an $O(\text{cpn}(G) \cdot n\log m(k(\log m + \log k \cdot \log\log m) + \log^2 m))$ time randomized algorithm for the k-mismatches problem over arbitrary M.

Matrix Multiplication Witnesses and All-Pairs k-Shortest Paths Alon, Galil, Margalit and Naor [3] and Seidel [24] independently discovered a simple randomized algorithm to solve the problem of finding a witness for Boolean matrix multiplication that runs in expected $O(M(n)\log n)$ time, where $M(n)$ is the time of the fastest algorithm for matrix multiplication. Alon and Naor [4] derandomized the algorithm, providing an $O(M(n) \cdot \text{polylog}(n))$ deterministic algorithm for this problem.

Using our construction we solve the k-witness problem for matrices in $O(M(n) \cdot k \cdot \text{polylog}(n))$ time by a deterministic algorithm, and in $O(M(n) \cdot (k(\log n + \log k \cdot \log\log n) + \log^2 n))$ time by a randomized algorithm. This yields a solution to the all-pairs k-shortest paths problem with the same time complexity as the k-witnesses for the Boolean matrix multiplication problem.

2 The Peeling Processes

2.1 Peeling Collections and the Peeling Process

Consider the k-reconstruction problem which was defined in the introduction. We are interested in finding k elements of each set S_i with minimal work. Thus, we seek an algorithm that both uses few calls to these procedures and allows for efficient reconstruction of the k elements based on the results obtained from these calls.

Definition 1. *For sets S and A we say that A* peels *S if $|S \cap A| = 1$. If $S \cap A = \{s\}$ we say that A peels the element s.*

Definition 2. *Let S be a set and F a collection of sets. We say that F is a k-*separator for *S if there are sets $A_1, \ldots, A_{\min(k,|S|)} \in F$ such that:*

1. *for each i, A_i peels S,*
2. *for $i \neq j$, $S \cap A_i \neq S \cap A_j$ (i.e. the sets peel different elements of S).*

*For a collection of sets $\mathcal{S} = \{S_1, \ldots, S_n\}$ we say that F is a k-*separator for *$\mathcal{S}$ if it is a k-separator for each $S \in \mathcal{S}$.*

Suppose that F is a k-separator for S. Then, it is possible to reconstruct $k' = \min(k, |S|)$ elements of S by simply invoking the procedure $\text{ISize}_S(A)$ for all $A \in F$. For each A such that $\text{ISize}_S(A) = 1$ compute $s_A = \text{ISum}_S(A)$. The element s_A is necessarily a member of S, and since F is a k-separator, there must be at least k' distinct s_A's.

A k-separator requires that for each element $s \in S$ to be recovered, there is a *distinct* set $A \in F$ that peels s. We now show how to reconstruct k' elements of S with a weaker condition. We do so by iteratively "peeling off" elements from S.

Let $P = (A_1, \ldots, A_k)$ be a sequence of sets. We say that P is a *peeling sequence for S* if for $i = 1, \ldots, k$, A_i peels $(S - \cup_{j=1}^{i-1} A_j)$. Let F be a collection of sets, and $P = (A_1, \ldots, A_k)$ a sequence of sets. We say that F *contains* P if $A_1, \ldots, A_k \in F$.

Definition 3. *Let S be a set and F a collection of sets. We say that F is a k-*peeler for S *if F contains a peeling sequence for S of length $k' = \min(k, |S|)$.*

*For a collection of sets $\mathcal{S} = \{S_1, \ldots, S_n\}$ we say that F is a k-*peeler for $\mathcal{S}$ *if it is a k-peeler for each $S \in \mathcal{S}$. We say that F is a* bounded k-peeler for $\mathcal{S}$ *if it is a k-peeler for every set $S \in \mathcal{S}$ of size at most k.*

Note that a k-separator for a set S is also a k-peeler for S. Let F be a k-peeler for S. Consider the following process, which we call *the peeling procedure*:

```
1   For each $A \in F$ assign $z_A \leftarrow \textsc{ISize}_S(A)$ and $m_A \leftarrow \textsc{ISum}_S(A)$
2   $\textsc{Ones} \leftarrow \{A \in F : z_A = 1\}$
3   While $\textsc{Ones}$ is not empty do
4       Choose any $A \in \textsc{Ones}$ and remove $A$ from $\textsc{Ones}$
5       Output $m_A$
6       For each $A' \in F$ such that $m_A \in A'$ do
7           $z_{A'} \leftarrow z_{A'} - 1$
8           If $z_{A'} = 1$ then add $A'$ to $\textsc{Ones}$
9           $m_{A'} \leftarrow m_{A'} - m_A$
```

At each iteration of lines 3–9, the procedure finds one element of S and "peels" it off S. In addition, the procedure also updates the corresponding values of the $\textsc{ISize}_S(A)$ and $\textsc{ISum}_S(A)$, for all $A \in F$.

We assume that the sets of F are represented by linked lists of the elements of the set. Thus, in time $O(\sum_{A \in F} |A|)$ we can create for every element $s \in U$ a list of all the sets of F that contain s, and use these lists for the loop in line 6.

Since F is a k-peeler for S, if the procedure happens to choose (in line 4) exactly the A's from the peeling sequence contained in F, and in exactly the right order, then the process will necessarily find at least $k' = \min(k, |S|)$ elements of S. What happens if other A's are chosen? Or in a different order? The following lemma proves that k' elements are necessarily found, regardless of the choice of the A's and their ordering.

Lemma 1. *Suppose that F is a k-peeler for S. Then the peeling procedure necessarily finds at least $k' = \min(k, |S|)$ elements of S.*

Proof. Let $(B_1, \ldots, B_{k'})$ be the peeling sequence for S contained in F. Consider an invocation of the peeling procedure. Let $P = (A_1, \ldots, A_t)$ be the sequence of sets chosen so far in the peeling process. We show that if $t < k'$ then P can necessarily be extended. For $j = 1, \ldots, k'$ let $b_j = (S - \cup_{i=1}^{j-1} B_i) \cap B_j$ (i.e. b_j is the element of S peeled by B_j) and for

$j = 1, \ldots, t$ let $a_j = (S - \cup_{i=1}^{j-1} A_i) \cap A_j$. Let j_0 be smallest index such that $b_{j_0} \notin \{a_1, \ldots, a_t\}$ (such an index necessarily exists since $t < k'$ and all b_j's are distinct). Then,

$$B_{j_0} \cap (S - \cup_{i=1}^{t} A_i) = \{b_{j_0}\}.$$

Hence, the peeling process can be continued by choosing B_{j_0}. ■

The peeling procedure can be extended to reconstruct a collection of sets $\mathcal{S} = \{S_1, \ldots, S_n\}$ with a k-peeler F for $\mathcal{S}$: In step 1 compute $\langle z_{A,1}, \ldots, z_{A,n} \rangle \leftarrow \text{ISIZE}_{S_1,\ldots,S_n}(A)$ and $\langle m_{A,1}, \ldots, m_{A,n} \rangle \leftarrow \text{ISUM}_{S_1,\ldots,S_n}(A)$. Then, perform steps 2–9 for every set S_i separately.

Two factors determine the time complexity of the peeling procedure. One is the size of the peeling collection F. The other is the total number of times lines 7–9 are performed during the peeling process. This number can be bounded as follows. For an element $u \in U$ and a peeling collection F, let $\text{OCC}_u(F)$ be the number of sets in F that contain u. Define $\text{MAXOCC}(F) = \max_{u \in U} \{\text{OCC}_u(F)\}$. The number of times lines 7–9 are performed when reconstructing a collection of n sets is bounded by $nk \cdot \text{MAXOCC}(F)$. Therefore, we obtain the following lemma (recall that a call to the procedures $\text{ISIZE}_{S_1,\ldots,S_n}(A)$ and $\text{ISUM}_{S_1,\ldots,S_n}(A)$ takes $O(f)$ steps).

Lemma 2. *The time complexity of the peeling procedure is $O(f \cdot |F| + nk \cdot \text{MAXOCC}(F))$.*

Hence, we seek small peeling collections with a small $\text{MAXOCC}(F)$. Clearly, for any set S of size k there is a trivial k-peeler with $|F| = k$ and $\text{MAXOCC}(F) = 1$. However, since we are not provided with S explicitly, we must construct the peeling collection without explicit knowledge of S.

We note that some of our definitions are similar to previous work. k-separators and k-peelers are similar to k-selective collections [7, 8, 18]. A collection F is *k-selective* for a collection $\mathcal{S}$ if it is a 1-peeler for every set $S \in \mathcal{S}$ with $|S| \leq k$. We also say that a collection F is *universal k-selective* if it is k-selective for the collection $\mathcal{S}_U$ of all subsets of U. Similarly, F is *universal bounded k-peeler* if it is a bounded k-peeler for $\mathcal{S}_U$. Constructions of universal k-selective collections were given in [8, 18]. A lower bound for k-selective collections in a non-universal setting was given in [7]. In the universal setting, a collection F is universal k-selective if and only if it is a universal bounded k-peeler[1]. However, in the non-universal setting the properties of being k-selective and being a bounded k-peeler are different: Every bounded k-peeler is also k-selective, but the converse need not be true.

Also, bounded k-peelers are similar to the universal lists in [20]. The universal lists are defined for the universal setting. As consequence, the construction is Section 2.3.2 is more efficient than the one given in [20].

[1]By definition, if F is a universal bounded k-peeler then F is universal k-selective. For the opposite direction, suppose that F is universal k-selective and let $S \subseteq U$ be a set of size at most k. There is a set $A_1 \in F$ that peels S. Since $S - A_1$ is of size at most $k-1$, there is a set $A_2 \in F$ that peels $S - A_1$. Repeating this process given a peeling sequence $(A_1, A_2, \ldots, A_{|S|})$ for S. Therefore, F is a universal bounded k-peeler.

2.2 Deterministic Constructions

2.2.1 Bounded k-Peelers

Consider the problem of constructing a universal bounded k-peeler, or equivalently, constructing a k-selective collection. Indyk [18] showed how to deterministically construct a universal bounded k-peeler.

Theorem 3 (Indyk [18]). *There is a deterministic algorithm that for every k constructs a universal bounded k-peeler F with $|F| = O(k \cdot \mathrm{polylog}(m))$ and $\textsc{MaxOcc}(F) = O(\mathrm{polylog}(m))$. The time complexity of the algorithm is $O(m \cdot \mathrm{polylog}(m))$.*

We note that there is no efficient construction of a universal (unbounded) k-peeler. More precisely, the size of a universal k-peeler must be at least m (even for $k = 1$) [11].

2.2.2 k-Separators

Let $\mathcal{S} = \{S_1, \dots, S_n\}$ be a collection of sets, each of size at least $ck \log^3 m$, for some constant c to be determined later. We show how to construct a k-separator for $\mathcal{S}$. The construction uses the algorithm of Alon and Naor [4]. Using the terminology presented here, Alon and Naor provided an algorithm to construct a 1-separator for $\mathcal{S}$. The algorithm runs in $O(f \log n \cdot \mathrm{polylog}(m))$ time and outputs a collection F of size $O(\log n \cdot \mathrm{polylog}(m))$.

Lemma 4. *There is a deterministic algorithm that for every k constructs a collection H of subsets of U that satisfies*

1. *For every $S \subseteq U$ such that $|S| \geq ck \log^3 m$, there are disjoint sets $A_1, \dots, A_k \in H$ such that $S \cap A_i$ is non-empty for all i, and*
2. *$|H| = O(k \cdot \mathrm{polylog}(m))$.*

The running time of the algorithm is $O(m \cdot \mathrm{polylog}(m))$.

Proof. The proof of the lemma is similar to the proof of Theorem 3.1 in [18]. From [25], there is an algorithm that given N and K, integer powers of two, constructs a bipartite graph $G = (A, B, E)$ such that

1. The degree of every vertex in A is D, where $D = O(\mathrm{polylog}(N))$.
2. $|A| = N$.
3. $\frac{KD}{c_0 \log^3 N} \leq |B| \leq KD$, where c_0 is a constant.
4. For every $X \subseteq A$ of size at least K, the number of vertices in B that are neighbors of vertices in X is at least $|B|/2$.

The construction of G takes $O(N \cdot \mathrm{polylog}(N))$ time.

Given U and k, build a graph $G = (A, B, E)$ using the above algorithm with $N = \alpha(m)$ and $K = \alpha(2c_0 \log^3 N \cdot k)$, where $\alpha(x) = 2^{\lceil \log_2 x \rceil}$. Set $c = 32c_0$. Assign a distinct element from A to each element $u \in U$. For each vertex $x \in A$, order its neighbors arbitrarily. Define mappings $h_1, \ldots, h_D$ from U to B as follows: $h_i(u)$ is the ith neighbor of the vertex x that was assigned to u. For a set $S \subseteq U$ define $h_i(S) = \{h_i(s) : s \in S\}$. Define H to be the collection of all sets $h_i^{-1}(b)$ for all $i \leq D$ and $b \in B$.

Clearly, $|H| = D \cdot |B| = O(k \cdot \mathrm{polylog}(m))$. We next show that H satisfies the first property of the lemma. Consider a subset S of U of size at least $ck \log^3 m$. We have that $K = \alpha(2c_0 k \log^3 N) \leq 4c_0 k \log^3(2m) \leq ck \log^3 m$. Therefore, $| \cup_{i=1}^{D} h_i(S)| \geq |B|/2$. By the pigeon-hole principle, there is an index i such that $|h_i(S)| \geq |B|/(2D) \geq K/(2c_0 \log^3 N) \geq k$. Let $b_1, \ldots, b_k$ be k distinct elements from $h_i(S)$. The sets $A_j = h_i^{-1}(b_j)$ for $j = 1, \ldots, k$ are disjoint and $S \cap A_j \neq \emptyset$ for all j. ■

To construct a k-separator for $S_1, \ldots, S_n$ do the following. Let H be the collection provided by Lemma 4. For every $A \in H$, use the Alon-Naor algorithm to construct a 1-separator for $S_1 \cap A, \ldots, S_n \cap A$. The union of these 1-separators is a k-separator for $S_1, \ldots, S_n$. We obtain:

Theorem 5. *There is a constant c such that the following holds. Let $\mathcal{S} = \{S_1, \ldots, S_n\}$ be a collection of sets each of size at least $ck \log^3 m$. There is a deterministic algorithm that constructs a k-separator collection F for $\mathcal{S}$ such that $|F| = O(k \log n \cdot \mathrm{polylog}(m))$. The construction of F takes $O(fk \log n \cdot \mathrm{polylog}(m))$ time.*

2.3 Randomized Constructions

2.3.1 Preliminaries

For $p \in [0, 1]$, build a subset of U by choosing each element of U independently at random with probability p. We call the resulting set a *p-random set*. Let $\alpha = 1/(2e)$.

Lemma 6. *Let A be a $1/t$-random set, and let S be a set with $t/2 \leq |S| \leq t$. Then,* $\Pr[A \textit{ peels } S] \geq \alpha$.

Proof. Let k denote $|S|$. Let X be the number of elements from S chosen to A. The random variable X has binomial distribution $N(k, 1/t)$. Hence,

$$\Pr[X = 1] = k\frac{1}{t}\left(1 - \frac{1}{t}\right)^{k-1} \geq \frac{k}{t}\left(1 - \frac{1}{t}\right)^{t-1} \geq \frac{1}{2}e^{-1}. \qquad ■$$

Consider a set S of size t and a collection of sets F. We say that F *peels S down to size t'* if F is a $(t - t')$-peeler for S.

Corollary 7. *Let F be a collection of r $1/t$-random sets, and let S be a set with $t/2 < |S| \leq t$. Then* $\Pr[F \textit{ does not peel } S \textit{ down to size } t/2] \leq 2^t(1 - \alpha)^r$.

Proof. For a set $S' \subseteq S$ of size greater than $t/2$, let $E_{S'}$ be the event that F does not peal S'. Clearly, if none of the events $E_{S'}$ occur, then F peals S down to size $t/2$. Thus, by Lemma 6,

$$\begin{aligned}\Pr\left[F \text{ does not peel } S \text{ down to size } t/2\right] &\le \Pr\left[\cup_{S'\subseteq S, |S'|>t/2} E_{S'}\right] \\ &\le \sum_{\substack{S'\subseteq S\\ |S'|>t/2}} \Pr\left[E_{S'}\right] \le 2^{|S|}(1-\alpha)^r \le 2^t(1-\alpha)^r. \quad \blacksquare\end{aligned}$$

In the next sections we use the following inequalities of Hoeffding [16].

Lemma 8 (Hoeffding's inequality). *Let $X_1, \ldots, X_n$ be independent random variables, with $0 \le X_i \le 1$ for all i, and let $X = \sum_{i=1}^n X_i$. Then, for every $a \ge E[X]$, $\Pr[X \ge E[X] + a] \le e^{-a/3}$, and for every $\delta > 0$, $\Pr[X \le (1-\delta)E[X]] \le e^{-\delta^2 E[X]/2}$.*

2.3.2 Bounded k-Peelers

Let $\mathcal{S}$ be a collection of n sets. We now show how to construct a collection F such that with probability $\ge 1 - \frac{1}{2}n^{-1}$, F is a bounded k-peeler for $\mathcal{S}$. For $k = 1$, the collection $\{U\}$ is a universal bounded 1-peeler, so assume for the rest of the section that $k \ge 2$. Construct F as a union of collections $F^{(j)}$, $j = 0, \ldots, \lceil \log k \rceil$ (throughout the paper, $\log x = \log_2 x$). $F^{(0)} = \{U\}$. For $j > 0$, the collection $F^{(j)}$ consists of r_j $\frac{1}{2^j}$-random sets, with $r_j = \frac{2^j + 3\log(nk)}{\log(1/(1-\alpha))}$ (to simplify the presentation, we omit the ceiling function in the definition of r_j).

Theorem 9. *Let $\mathcal{S}$ be a collection of n sets. With probability $\ge 1 - \frac{1}{2}n^{-1}$, F is a bounded k-peeler for $\mathcal{S}$, and $\text{MaxOcc}(F) = O(\log(nm))$. The size of F is $O(k + \log k \cdot \log n)$, and the time complexity for constructing F is $O(m \log(nk))$.*

Proof. Consider some set $S \in \mathcal{S}$ of size $k' \le k$, and let $S^{(\lceil \log k' \rceil)}$ denote S. Use $F^{(\lceil \log k' \rceil)}$ to peel $S^{(\lceil \log k' \rceil)}$ down to a set of size $2^{\lceil \log k' \rceil - 1}$. Denote the resulting set by $S^{(\lceil \log k' \rceil - 1)}$. Use $F^{(\lceil \log k' \rceil - 1)}$ to peel $S^{(\lceil \log k' \rceil - 1)}$ down to size $2^{\lceil \log k' \rceil - 2}$. Denote the resulting set by $S^{(\lceil \log k' \rceil - 2)}$. Continue in this way until the entire set S is peeled, or until $F^{(j)}$ fails to peel $S^{(j)}$ down to size 2^{j-1} for some j. Assuming that S was peeled successfully to $S^{(j)}$, by Corollary 7, the probability that $F^{(j)}$ fails to peel $S^{(j)}$ down to size 2^{j-1} is $\le 2^{2^j}(1-\alpha)^{r_j} = (nk)^{-3}$. Thus, the probability that F is not a bounded k-peeler for S is $\le \sum_{j=1}^{\lceil \log k' \rceil} \Pr\left[F^{(j)} \text{ fails to peel } S^{(j)} \text{ down to size } 2^{j-1}\right] \le \lceil \log k' \rceil \cdot (nk)^{-3} \le \frac{1}{4}n^{-2}$. Therefore, the probability that F is not a bounded k-peeler for $\mathcal{S}$ is $\le \frac{1}{4}n^{-1}$.

Next we bound $\text{MaxOcc}(F)$. We have that

$$E[\text{Occ}_u(F)] = 1 + \sum_{j=1}^{\lceil \log k \rceil} 2^{-j} \frac{2^j + 3\log(nk)}{\log(1/(1-\alpha))} \le 1 + \frac{\lceil \log k \rceil + 3\log(nk)}{\log(1/(1-\alpha))} \le 20\log(nk).$$

Hence, by Hoeffding's inequality,

$$\begin{aligned}\Pr\left[\text{Occ}_u(F) \ge 40\log(nm)\right] &\le \Pr\left[\text{Occ}_u(F) \ge E[\text{Occ}_u(F)] + 20\log(nm)\right] \\ &\le e^{-20\log(nm)/3} \le \frac{1}{4}(nm)^{-1}.\end{aligned}$$

There are m different u's, so $\Pr[\textsc{MaxOcc}(F) \geq 40\log(nm)] \leq \frac{1}{4}n^{-1}$.

The size of F is

$$|F| = 1 + \sum_{j=1}^{\lceil \log k \rceil} \frac{2^j + 3\log(nk)}{\log(1/(1-\alpha))} = O(k + \log k \cdot \log(nk)) = O(k + \log k \cdot \log n).$$

The time complexity for building F by a straightforward algorithm is $O(|F| \cdot m) = O((k + \log k \cdot \log n)m)$. We now describe a faster construction algorithm. For every $u \in U$, the number of sets in $F^{(j)}$ that contain u has binomial distribution $N(r_j, 2^{-j})$. Therefore, to build $F^{(j)}$, create r_j empty sets $A_1^{(j)}, \ldots, A_{r_j}^{(j)}$. For every $u \in U$, randomly select a value d_u from the distribution $N(r_j, 2^{-j})$. Then, randomly select d_u distinct sets from $A_1^{(j)}, \ldots, A_{r_j}^{(j)}$, and add u to the selected sets. The expected running time of this algorithm is $O(|F| + E[\textsc{Occ}_u(F)] \cdot m) = O(m \log(nk))$. ■

2.3.3 k-Separators

Let $\mathcal{S}$ be a collection of n sets, each of size $\geq 4k$. We show how to construct a collection F such that with probability $\geq 1 - \frac{1}{2}n^{-1}$, F is a k-separator for $\mathcal{S}$. For $j = \lceil \log(4k) \rceil, \ldots, \lceil \log m \rceil$, let $F^{(j)}$ be a collection of r_j $\frac{1}{2^j}$-random sets, with $r_j = \frac{16}{\alpha}\ln(2n) + \frac{2}{\alpha}jk/(j - 1 - \log k)$. Let $F = \bigcup_{j=\lceil \log(4k) \rceil}^{\lceil \log m \rceil} F^{(j)}$.

Theorem 10. *Let $\mathcal{S}$ be collection of n sets, each of size $\geq 4k$. With probability $\geq 1 - \frac{1}{2}n^{-1}$, F is a k-separator for $\mathcal{S}$. The size of F is $O(k(\log m + \log k \cdot \log\log m) + \log n \cdot \log m)$, and the time complexity for constructing F is $O(m\log(nmk) + k\log k \cdot \log\log m)$.*

Proof. Consider a set $S \in \mathcal{S}$, and let j be an integer such that $2^{j-1} < |S| \leq 2^j$. Note that $|S| \geq 4k$, so $j \geq \lceil \log(4k) \rceil$. Let X be the number of sets in $F^{(j)}$ that peel S. By Lemma 6, the probability that a fixed set $A \in F^{(j)}$ peels S is at least α. Hence, $E[X] \geq \alpha r_j$. By Hoeffding's inequality,

$$\Pr\left[X \leq \frac{1}{2}\alpha r_j\right] \leq \Pr\left[X \leq \frac{1}{2}E[X]\right] \leq e^{-E[X]/8} \leq e^{-\alpha r_j/8} \leq \frac{1}{4n^2}.$$

For an element $s \in S$, we say that s is *peeled* if there is a set in F that peels s. Let P be the set of peeled elements in S. A set in F^j is equally likely to peel any element of S and the sets in F^j are all independent. Therefore, for $S' \subseteq S$, $\Pr[P \subseteq S' | X = x] = (|S'|/|S|)^x$. Therefore, assuming that $X > \frac{1}{2}\alpha r_j$,

$$\Pr[|P| < k] \leq \sum_{\substack{S' \subseteq S \\ |S'| = k-1}} \Pr[P \subseteq S'] \leq |S|^{k-1}\left(\frac{k-1}{|S|}\right)^{\frac{1}{2}\alpha r_j} \leq 2^{jk}\left(\frac{k}{2^{j-1}}\right)^{\frac{1}{2}\alpha r_j} \leq \frac{1}{4n^2}.$$

Thus, the probability that F is not a k-separator for S is $\leq \frac{1}{2}n^{-2}$. Hence, the probability that F is not a k-separator of $\mathcal{S}$ is $\leq \frac{1}{2}n^{-1}$.

The bound on $|F|$ stated in the theorem is straightforward. The collection F can be build in $O(m\log(nmk)+k\log k\cdot\log\log m)$ time using the same approach described in the proof of Theorem 9 (we can give a smaller bound on the construction time, but the given bound suffices for our purpose). ■

2.4 Solving the k-Reconstruction Problem

We are now ready to show how to use the separating and peeling collections to solve the reconstruction problems presented in the introduction. Recall that given a collection $\mathcal{S}$ containing n subsets of a set U of size m, the *k-reconstruction problem* is: For each $S\in\mathcal{S}$, find $\min(k,|S|)$ elements of S. The *bounded k-reconstruction problem* is: Fully reconstruct every set $S\in\mathcal{S}$ of size at most k. We now give algorithms solving these problems.

Theorem 11. *Suppose that computing* $\text{ISize}_S(A)$ *and* $\text{ISum}_S(A)$ *for all* $S\in\mathcal{S}$ *and one set* A *takes* $O(f)$ *steps. Then, there are deterministic and randomized algorithms for the bounded k-reconstruction problem with the following running times:*

- *Deterministic:* $O(fk\cdot\text{polylog}(m))$ *steps.*
- *Randomized:* $O(f(k+\log k\cdot\log n)+nk\log(nm))$ *steps.*

Proof. Given an instance $\mathcal{S}$ of the bounded k-reconstruction problem, construct a bounded k-peeler for $\mathcal{S}$ using the deterministic construction of Theorem 3 or the randomized construction of Theorem 9. Then use the peeling procedure. By Lemma 2, the time complexity of the peeling procedure is $O(f\cdot|F|+nk\cdot\text{MaxOcc}(F))$. In the deterministic construction, $|F|=O(k\cdot\text{polylog}(m))$, $\text{MaxOcc}(F)=O(\text{polylog}(m))$, and the time complexity for constructing F is $O(m\cdot\text{polylog}(m))$. Therefore, the time complexity of the deterministic algorithm is

$$O(m\cdot\text{polylog}(m)+fk\cdot\text{polylog}(m)+nk\cdot\text{polylog}(m))=O(fk\cdot\text{polylog}(m)),$$

where the equality follows from the assumption that $f=\Omega(m+n)$.

In the randomized construction, $|F|=O(k+\log k\cdot\log n)$, $\text{MaxOcc}(F)=O(\log(nm))$, and the time complexity for constructing F is $O(m\log(nk))$. Thus, the time complexity of the randomized algorithm is

$$O(m\log(nk)+f(k+\log k\cdot\log n)+nk\cdot\log(nm))=O(f(k+\log k\cdot\log n)+nk\log(nm)).\ ■$$

The randomized algorithm solves the reconstruction problem with probability at least 1/2. One can repeatedly run the algorithm until all sets are reconstructed, and the above result second bound is the expected number of steps.

Theorem 12. *Suppose that computing* $\text{ISize}_S(A)$ *and* $\text{ISum}_S(A)$ *for all* $S\in\mathcal{S}$ *and one set* A *takes* $O(f)$ *steps. Then, there are deterministic and randomized algorithms for the k-reconstruction problem with the following running times:*

- *Deterministic:* $O(fk \log n \cdot \operatorname{polylog}(m))$ *steps.*
- *Randomized:* $O(f(k(\log m + \log k \cdot \log\log m) + \log n \cdot \log m) + nk \log n)$ *steps.*

Proof. Choose some integer k', and construct both a bounded k'-peeler, which will be used to fully reconstruct sets with at most k' elements (using the peeling procedure), and a k-separator, which will be used to reconstruct k elements from sets with at least k' elements. For the deterministic algorithm choose $k' = ck \log^3 m$ (where c is the constant from Theorem 5), and use Theorem 3 for the bounded k'-peeler and Theorem 5 for the k-separator. As in the proof of Theorem 11, the time complexity for building the bounded k'-peeler and using the peeling procedure is $O(fk' \cdot \operatorname{polylog}(m)) = O(fk \cdot \operatorname{polylog}(m))$. By Theorem 5, the time complexity for building the k-separator is $O(fk \log n \cdot \operatorname{polylog}(m))$, and the time complexity for using the k-separator to reconstruct elements is $O(f \cdot |F|) = O(fk \log n \cdot \operatorname{polylog}(m))$. Therefore, the time complexity of the deterministic algorithm is $O(fk \log n \cdot \operatorname{polylog}(m))$

For the randomized algorithm choose $k' = 4k$, and use Theorem 9 for the bounded k'-peeler and Theorem 10 for the k-separator. The time complexity for building the bounded k'-peeler and using the peeling procedure is $O(f(k+\log k \cdot \log n)+nk \log(nm))$. By Theorem 10, the time complexity for building the k-separator is $O(m \log(nmk) + k \log k \cdot \log\log m)$, and the time complexity for using the k-separator to reconstruct elements is

$$O(f \cdot |F|) = O(f(k(\log m + \log k \cdot \log\log m) + \log n \cdot \log m)).$$

By summing the bounds above and using the assumption that $f = \Omega(m+n)$, we obtain the bound stated in the theorem. ■

We note that except for the deterministic algorithm for the (unbounded) k-reconstruction problem, the other algorithms are *non-adaptive*, in the sense that the set of A's on which the procedures $\textsc{ISize}_S(A)$ and $\textsc{ISum}_S(A)$ are performed is determined independently of the outcomes of any other such calls. The deterministic algorithm for the k-reconstruction problem is adaptive due to the adaptive algorithm of Alon and Naor.

3 Applications

3.1 Matrix Multiplication Witnesses and All-Pairs k-Shortest Paths

Consider the undirected unweighted shortest paths problem. Let $G = (V, E)$ be some undirected graph, where $V = \{0, \ldots, n-1\}$. We use $\delta(u, v)$ to denote the length of the shortest path between vertices u and v. Let $p_H(u, v)$ denote the number of paths from u to v in a graph H.

For a vertex v from G, we define the *v-successor graph* of G, denoted G_v, to be a directed graph with vertex set V, which contains an edge (u, w) for every pair of vertices $u, w \in V$ such that there is a path from u to v of length $\delta(u, v)$ that contains the edge (u, w). The following claim is trivial.

Claim 13. *For every graph G and a vertex v in G,* *1. G_v is acyclic.*

2. Every path from u to v in G_v is a shortest path from u to v in G, and vice versa.

3. Given G_v, $l \leq p_{G_v}(u,v)$ different paths from u to v in G_v can be reported in time $O(l \cdot n)$.

A *k-subgraph* of a directed graph H is a subgraph H' of H such that for every vertex v in H, if v has l outgoing edges in H, then H' contains exactly $\min(k,l)$ of these edges.

Lemma 14. *If G' is a k-subgraph of G_v then for every vertex u, $p_{G'}(u,v) \geq \min(k, p_{G_v}(u,v))$.*

Proof. Let G' be some k-subgraph of G_v. We prove the lemma using induction on the distance between u and v. If $u = v$ or u is a neighbor of v then the lemma is trivially true. Suppose that we proved the lemma for all vertices with a distance of at most $d-1$ from v. Let u be a vertex with $\delta(u,v) = d$. Let $u_1, \ldots, u_r$ be the neighbors of u in G_v. If $r \geq k$, then u has k neighbors in G'. For each such neighbor u_i, by the induction hypothesis there is at least one path in G' from u_i to v. Therefore, $p_{G'}(u,v) \geq k$.

If $r < k$ then $u_1, \ldots, u_r$ are also neighbors of u in G'. Hence, by the induction hypothesis,

$$p_{G'}(u,v) = \sum_{i=1}^{r} p_{G'}(u_i,v) \geq \sum_{i=1}^{r} \min(k, p_{G_v}(u_i,v)).$$

If $p_{G_v}(u_i,u) \geq k$ for some i then clearly $p_{G'}(u,v) \geq k$. Otherwise, $p_{G'}(u,v) = \sum_{i=1}^{r} p_{G_v}(u_i,v) = p_{G_v}(u,v)$. ■

Seidel [24] gave an algorithm for computing $\delta(u,v)$ for every pair of vertices u,v in G in time $O(M(n)\log n)$, where $M(n)$ is the time needed to multiply two n-by-n matrices. Moreover, he showed that (u,w) is an edge of G_v if and only if w is a neighbor of u and $\delta(w,v) \equiv \delta(u,v) - 1 \pmod 3$. Therefore, the algorithm for the k-shortest paths problem is as follows.

1. Compute $\delta(u,v)$ for all u and v.

2. Let B be the adjacency matrix of G, and let $C^{(i)} = \{c^{(i)}_{u,v}\}$ for $i = 0,1,2$, be Boolean matrices such that $c^{(i)}_{u,v} = 1$ if and only if $\delta(u,v) \equiv i - 1 \pmod 3$.

3. Solve the k-witnesses problem for the products $BC^{(i)}$, $i = 0,1,2$.

4. For every v, set $G_v = (V, E_v)$, where E_v is defined as follows. For every vertex $u \in V$ such that position u,v in the product $BC^{(\delta(u,v) \bmod 3)}$ is 1, E_v contains edges (u,w) for the up to k witnesses w for the 1 in position u,v in the product, as found in (3).

5. For every u and v, report $\min(k, p_{G_v}(u,v))$ different paths in G_v.

To solve the k-witnesses problem for the product $BC^{(i)}$, set $U = V$ to be the universe. Define sets $S_0, \ldots, S_{n^2-1}$ as follows. For every pair of vertices u and v, S_{un+v} is the set of all vertices w such that $b_{u,w} = 1$ and $c^{(i)}_{w,v} = 1$. In oder to compute $\textsc{ISize}_{S_0,\ldots,S_{n^2-1}}(A)$ and $\textsc{ISum}_{S_0,\ldots,S_{n^2-1}}(A)$ for some $A \subseteq U$, define $C = \{c_{w,v}\}$, where $c_{w,v} = 1$ if $w \in A$ and $c^{(i)}_{w,v} = 1$, and $c_{w,v} = 0$ otherwise. Also define $C' = \{c'_{w,v}\}$, where $c'_{w,v} = w$ if $c_{w,v} = 1$, and $c'_{w,v} = 0$ otherwise. Then, $\textsc{ISize}_{S_{un+v}}(A)$ and $\textsc{ISum}_{S_{un+v}}(A)$ are the elements at position u, v in the products BC and BC', respectively.

The correctness of the algorithm follows from Claim 13 and Lemma 14. By Theorem 12, the time complexity is $O(M(n) \cdot k \cdot \text{polylog}(n))$ for the deterministic algorithm and $O(M(n) \cdot (k(\log n + \log k \cdot \log\log n) + \log^2 n))$ for the randomized algorithm.

3.2 Aligned Ones and Matches

Let a text $t = t_0 \cdots t_{n-1}$ and a pattern $p = p_0 \cdots p_{m-1}$ be the input for the k-aligned ones problem. Set $U = \{0, \ldots, m-1\}$ to be the universe. For each location i of t define $S_i \subseteq U$ to be the set of all indices j such that $p_j = t_{i+j} = 1$. We can assume w.l.o.g. that $n \leq 2m$, since if $n > m$, we can split t into pieces of length at most $2m$ with an overlap of $m-1$ characters between consecutive pieces.

To solve the k-aligned ones problem we need to reconstruct every set S_i with size at most k. This is exactly the setting of the peeling procedure and all that is needed is to describe how to compute the cardinality and sum functions, $\textsc{ISize}_{S_1,\ldots,S_n}(A)$, $\textsc{ISum}_{S_1,\ldots,S_n}(A)$.

One of the widely used methods in pattern matching problems is the convolution method [14]. This method was also used in [5, 23]. We use this method here as well. Let $a[0 \ldots n-1]$ and $b[0 \ldots m-1]$ be vectors of elements over a semiring R with operations $(\oplus, \otimes)$. The *convolution* of a and b is defined as a vector $a * b$ such that $(a * b)[i] = \oplus_{j=0}^{m-1}(a[i-j] \otimes b[j])$ for $0 \leq i \leq n-1$ (out of range values are assumed to be 0). We also define $a \circ b$ to be the closely related array $(a \circ b)[i] = \oplus_{j=0}^{m-1}(a[i+j] \otimes b[j])$ where $-m+1 \leq i \leq n-m$ (out of range values are assumed to be 0). It is easy to see that this operation and convolutions can be computed within the same time bound. We will be interested in *polynomial convolutions*, namely, where $R = (+, \times)$ is over a field, in our case $\Re$.

We use convolutions as follows. To compute $\textsc{ISize}_{S_1,\ldots,S_n}(A)$ define the vector a to be t and set $b[i]$ to be 1 if $i \in A$ and $p_i = 1$, and 0 otherwise. It is easy to see that $(a \circ b)[i] = \textsc{ISize}_{S_i}(A)$. To compute $\textsc{ISum}_{S_1,\ldots,S_n}(A)$ define the vector a to be t and set $b'[i]$ to be i if $b[i] = 1$, and 0 otherwise. Once again, it is easy to see that $(a \circ b')[i] = \textsc{ISum}_{S_i}(A)$.

Each of the two convolutions can be computed in $O(n \log m)$ time [2]. Hence, by Theorem 11 it follows that there is an $O(nk \cdot \text{polylog}(m))$ time deterministic algorithm and an $O(n \log m(k + \log k \cdot \log m))$ time randomized algorithm for the k-aligned ones problem.

The k-matches can be solved with the same reductions assuming that the alphabet of the text and pattern is binary. When Σ, the alphabet of p and t, is larger one can simply invoke the k-matches for binary alphabet procedure Σ times, once for each $a \in \Sigma$. By Theorem 12, this yields an $O(|\Sigma| nk \cdot \text{polylog}(m))$ deterministic algorithm and an $O(|\Sigma| n \log m \cdot (k(\log m + \log k \cdot \log\log m) + \log^2 m))$ randomized algorithm.

3.3 Mismatches under General Matching Relations

In [23, Theorem 5] it was shown that the k-mismatches problem over arbitrary M can be solved with $\text{cpn}(G)$ calls to the k-witnesses problem over binary alphabets (see Section 1.2). The k-witnesses problem can be transformed into a k-matches problem by complementing the (binary) pattern p and retaining the text t as is. Obviously, every mismatch pair is now a match pair. Hence, the k-mismatches problem can be solved with an $O(\text{cpn}(G) \cdot nk \cdot \text{polylog}(m))$ time deterministic algorithm and an $O(\text{cpn}(G) \cdot n \log m \cdot (k(\log m + \log k \cdot \log\log m) + \log^2 m))$ time randomized algorithm.

References

[1] K. Abrahamson. Generalized string matching. *SIAM J. on Computing*, 16(6):1039–1051, 1987.

[2] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974.

[3] N. Alon, Z. Galil, O. Margalit, and M. Naor. Witnesses for boolean matrix multiplication and for shortest paths. *Proc. 33rd Symposium on Foundations of Computer Science (FOCS)*, pages 417–426, 1992.

[4] N. Alon and M. Naor. Derandomization, witnesses for boolean matrix multiplication and construction of perfect hash functions. *Algorithmica*, 16:434–449, 1996.

[5] A. Amir and M. Farach. Efficient 2-dimensional approximate matching of half-rectangular figures. *Information and Computation*, 118(1):1–11, 1995.

[6] A. Amir, M. Lewenstein, and E. Porat. Faster algorithms for string matching with k mismatches. *J. of Algorithms*, 50(2):257–275, 2004.

[7] C. Brito, E. Gafni, and S. Vaya. An information theoretic lower bound for broadcasting in radio networks. In *Proc. 21st Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 534–546, 2004.

[8] B. S. Chlebus, L. Gasieniec, A. Gibbons, A. Pelc, and W. Rytter. Deterministic broadcasting in ad hoc radio networks. *Distributed Computing*, 15(1):27–38, 2002.

[9] A. E. F. Clementi, A. Monti, and R. Silvestri. Selective families, superimposed codes, and broadcasting on unknown radio networks. *Proc. 13th Symposium on Discrete Algorithms (SODA)*, pages 709–718, 2001.

[10] P. Clifford and R. Clifford. Simple deterministic wildcard matching. *Information Processing Letters*, 101(2):53–54, 2007.

[11] P. Damaschke. Randomized group testing for mutually obscuring defectives. *Information Processing Letters*, 67:131–135, 1998.

[12] I. Dinstein, G. M. Landau, and G. Guy. Parallel (PRAM EREW) algorithms for contour-based 2D shape recognition. *Pattern Recognition*, 24(10):929–942, 1991.

[13] D. Z. Du and F. K. Hwang. *Combinatorial group testing and its applications.* World Scientific, 2000.

[14] M. J. Fischer and M. S. Paterson. String matching and other products. In *Proc. 7th SIAM-AMS Complexity of Computation*, pages 113–125, 1974.

[15] Z. Galil and R. Giancarlo. Improved string matching with k mismatches. *SIGACT News*, 17(4):52–54, 1986.

[16] W. Hoeffding. Probability inequalities for sums of bounded random variables. *J. Amer. Statist. Assoc.*, 58:13–30, 1963.

[17] P. Indyk. Interpolation of symmetric functions and a new type of combinatorial design. In *Proc. 31st Symposium on Theory of Computing (STOC)*, pages 736–740, 1999.

[18] P. Indyk. Explicit constructions of selectors and related combinatorial structures, with applications. In *Proc. 13th Symposium on Discrete Algorithms (SODA)*, pages 697–704, 2002.

[19] D. E. Knuth, J. H. Morris, and V. R. Pratt. Fast pattern matching in strings. *SIAM J. on Computing*, 6:323–350, 1977.

[20] J. Komlós and A. G. Greenberg. An asymptotically nonadaptive algorithm for conflict resolution in multiple-access channels. *IEEE Trans. on Information Theory*, 31(2):302–306, 1985.

[21] G. M. Landau and U. Vishkin. Efficient string matching with k mismatches. *Theoretical Computer Science*, 43:239–249, 1986.

[22] S. Muthukrishnan. Personal communication.

[23] S. Muthukrishnan. New results and open problems related to non-standard stringology. In *Proc. 6th Symposium on Combinatorial Pattern Matching (CPM)*, pages 298–317, 1995.

[24] R. Seidel. On the all-pairs-shortest-path problem in unweighted undirected graphs. *J. of Computer and System Sciences*, 51:400–403, 1995.

[25] A. Ta-Shma, C. Umans, and D. Zuckerman. Loss-less condensers, unbalanced expanders, and extractors. In *Proc. 33rd Symposium on the Theory of Computing (STOC)*, pages 143–152, 2001.