Flipping Letters to Minimize the Support of a String

Giuseppe Lancia, Franca Rinaldi, and Romeo Rizzi

Dipartimento di Matematica e Informatica
Via delle Scienze 206
33100 Udine, Italy
lancia,rinaldi,rizzi@dimi.uniud.it

Abstract. We consider a problem defined on strings and inspired by the way DNA encodes amino-acids as triplets of nucleotides. Given a string s on an alphabet Σ, a word-length k and a budget D, we want to determine the smallest number of distinct k-mers that can be left in s, if we are allowed to replace up to D letters of s. This problem has several parameters, and we discuss its complexity under all sorts of restrictions on the parameters values. We prove that some versions of the problem are polynomial, while the others are NP-hard.

Keywords: De Bruijn graphs, codons, string algorithms, parametrized complexity.

1 Introduction

In the problem studied in this paper, we consider a string s, of length n, over some alphabet Σ. For $1 \leq k \leq n$, we call a string of length k a k-mer. Note that there are altogether $|\Sigma|^k$ possible k-mers, and that s possesses (as substrings) at most $n-k+1$ *distinct* k-mers. Trivially, there are strings exhibiting only one k-mer (e.g., $s = 00000000$), while $s = 1110100011$ is a shortest string exhibiting all possible binary 3-mers (for results about the the problem of building a shortest string which possesses all possible k-mers see [2, 6, 5]).

Now, suppose we are allowed to replace up to D letters in s, so as to obtain a new string t. What is the smallest possible number of distinct k-mers that can be left in t?

For instance, assume $\Sigma = \{0, 1\}$ and

$$s = 011010011.$$

There are 6 different 3-mers in s, namely, 011, 110 101, 010, 100, 001. If we are allowed to flip up to $D = 2$ letters, we can obtain the new string

$$t = 01\mathbf{0}01001\mathbf{0},$$

which has only 3 distinct 3-mers, i.e., 010, 100 and 001.

The problem of flipping the right bits, so as to destroy the largest possible number of k-mers (i.e., to leave the fewest possible number of them) has the natural appeal for combinatorial mathematicians, in that it is a cute and challenging combinatorial question. We were inspired to study this problem by considering the way genes encode proteins in an organism's genome [4]. We will briefly discuss the situation here, but, before doing so, we remark that the applications of this paper's results to biology are just marginal. Our interest in the problem is purely on its theoretical aspects, and we will not focus on its practical applications.

A *gene* can be seen as a string over the alphabet {A,C,G,T}. The letters {A,C,G,T} are called *nucleotides*. Each substring of 3 consecutive nucleotides is called a *codon*, as it encodes for a particular *amino-acid*. The amino-acids are the basic constituents of *proteins*, and a protein is a chain of amino-acids, determined by the gene's sequence. A gene has 3 possible *(open) reading frames* (ORFs). I.e., depending on where we start to read, we may obtain a particular set of codons. At ORF number i, for $i = 1, 2, 3$, one reads all the codons that start at positions j, where $(j = i) \mod 3$. For instance, the 3 ORFs of the gene TACAGATAAGATACA are as follows:

```
       T A C A G A T A A G A T A C A
ORF₁   ↑     ↑     ↑     ↑     ↑
ORF₂     ↑     ↑     ↑     ↑     ↑
ORF₃       ↑     ↑     ↑     ↑     ↑
```

giving rise to the following codons: $ORF_1 = \{TAC, AGA, TAA, GAT, ACA\}$, $ORF_2 = \{ACA, GAT, AAG, ATA\}$, $ORF_3 = \{CAG, ATA, AGA, TAC\}$. Altogether, the distinct codons that we see in this gene are

$$\{TAC, AGA, TAA, GAT, ACA, AAG, ATA, CAG\}.$$

The number of distinct amino-acids a protein consists of is related to its complexity. Hence, we can consider a "complex" gene as one which shows the use of a large number of distinct codons (therefore relating the number of codons to the information content of the gene). Since DNA undergoes random mutations during time, the change of some of the nucleotides in the gene may result in a new sequence which displays much fewer codons than it originally did. Hence, we are led to formulate the question of how few different codons can still be present, in the worst case, after a certain number of mutations have taken place.

1.1 Notation and paper organization

Let s be the input string over an alphabet Σ. We denote by $n = |s|$ the length of s, and by $\sigma = |\Sigma|$ the alphabet size. For a given $k \in \mathbf{N}$, we denote by $\mathcal{K} = \Sigma^k$ the set of all k-mers over Σ. Furthermore, for a string x, we define its *support*, denoted by $K(x) \subseteq \mathcal{K}$, as the set of all k-mers which are substrings of s. For a string x, and $1 \le i \le j \le |x|$, we denote by $x[i, \cdots, j]$ the substring of x from position i to position j. If $i = j$, we shorthand $x[i] = x[i, \cdots, i]$ to represent the i-th character of x. Finally, given two strings x and y of the same length, we denote by $d_H(x, y)$ their *Hamming distance*, i.e., the number of positions in which they differ.

The problem studied in this paper can be stated as follows:

> "KMER": given $s \in \Sigma^n$, a length $k \in \mathbf{N}$ for k-mers, and a budget $D \in \mathbf{N}$, find a string $t \in \Sigma^n$ such that $d_H(s, t) \le D$ and $|K(t)|$ is minimum.

Notice that this problem has four parameters:

1. The string length n;
2. The alphabet size σ;
3. The k-mer length k;
4. The budget D.

	σ FREE k FREE	σ BOUNDED k FREE	σ FREE k BOUNDED	σ BOUNDED k BOUNDED
n FREE D FREE	1 NP-hard	2 NP-hard for $\sigma = 2$	3 NP-hard for $k = 2$	4 POLY $O(n)$
n BOUNDED D FREE	5 IMP.	6 IMP.	7 IMP.	8 IMP.
n FREE D BOUNDED	9 POLY $O(k\,\sigma^D\,n^{D+1})$	10 POLY $O(k\,n^{D+1})$	11 POLY $O(\,\sigma^D\,n^{D+1})$	12 POLY $O(n^{D+1})$
n BOUNDED D BOUNDED	13 IMP.	14 IMP.	15 IMP.	16 $O(1)$

Table 1. Parameterized complexity of "KMER"

In the remainder of the paper we will address the complexity of the problem when one or more of these parameters are limited to take some bounded values. For instance, when reasoning about genes and DNA sequences, it is $\sigma = 4$ and $k = 3$.

In Section 2 we characterize all possible cases for (n, σ, k, D), classifying them as either polynomial or NP-hard. In Section 3 we consider the polynomial cases of the problem "KMER", arising when D is bounded (as shown in Section 3.1) and when σ and k are both bounded (as shown in Section 3.2). In Section 4 we address the NP-hard cases of the problem, which happen when n and D are unbounded and at most one of σ and k is bounded. Finally, we draw some conclusions in Section 5.

2 Parameterized complexity

The problem has four parameters: the string length n, the alphabet size σ, the k-mer size k, and the budget D. We may consider all possibilities in which some of the parameters are bounded by constants (denoted as "BOUNDED" in Table 1), and some depend on the input (denoted as "FREE" in Table 1). All the cases are described in Table 1.

Before analyzing the cases, we make the following remark:

Remark 1. We can always assume
(i) $D \leq n$;
(ii) $k \leq n$;
(iii) $\sigma \leq n$.

Proof. Remarks (*i*) and (*ii*) are obvious. As for Remark (*iii*), we can reason as follows. Let α be any symbol occurring in s. Let t' be obtained from t by replacing with α each symbol which does not occur in s. It is not difficult to see that $d_H(s, t') \leq d_H(s, t)$ and $|K(t')| \leq |K(t)|$. Hence, t' is an optimal solution as well. Therefore, we can restrict Σ to the symbols originally in s and hence $\sigma \leq n$. □

We have the following situation:

- Cells 1, 2, 3: These cases are NP-hard, as shown in the Section 4 (in particular, the problem is NP-hard even for $\sigma = 2$ or for $k = 2$).

– Cell 4: This case is polynomial, as described in Section 3.2. The complexity is $O(2^{\sigma^k}\sigma^{k+1}n) = O(n)$.
– Cells 5, 6, 7, 8: Because of Remark 1(i), these cases are impossible (denoted by "IMP." in the table).
– Cells 9, 10, 11, 12: All these cases are polynomial, as described in Section 3.1. The intuition is that, in these cases, there are only a polynomial number of possible solutions, which can be checked exhaustively.
– Cells 13, 15: Because of Remark 1(iii) these are impossible cases (when n is bounded by a constant, also σ is bounded by a constant).
– Cell 14: This case is impossible by Remark 1(ii).
– Cell 16: This case is trivial. There is only a finite number of possible problem instances.

3 Polynomial cases

3.1 The case for D fixed

Theorem 2. *When D is bounded by a constant, while n is free, the problem* "KMER" *can be solved in polynomial time.*

Proof. Notice that there are $\binom{n}{D} = O(n^D)$ possible choices for the letters of s to flip, and σ^D possible ways to flip each one. Hence, there are only $O(\sigma^D n^D)$ possible solutions, which is $O(n^D)$ when σ is fixed, and $O(n^{2D})$ otherwise (because of Remark 1(iii)). Since there are only a polynomial number of solutions, and each solution value can be clearly computed in polynomial time (see Remark 3 here below), the enumeration of all possible solutions solves "KMER" in polynomial time. □

Remark 3. Given a string t of length n, the number of different k-mers occurring in t can be computed in $O(kn)$ time.

Proof. To count the k-mers, we scan the string t from left to right, and insert each k-mer in a trie T, initially empty. The branches of T are associated to the symbols of Σ and each leaf of T will represent a k-mer of t. A k-mer describes a path in T to a (possibly non-existing) leaf. Each time a new k-mer x is inserted in T, the path is followed in T up to the point p where it is no longer possible. If this happens at a leaf, then x was already present in T. Otherwise, p is an internal node. From p, we create the new branches that will lead to a new leaf (corresponding to x), and we increase a leaf counter (which eventually counts all k-mers). Notice that the insertion in T has cost $O(k)$. □

The complexity of the algorithms for all cases when D is fixed are reported in Table 1, cells 9–12.

3.2 The case for σ and k fixed

In this section we prove that, when k and σ are both bounded, the problem "KMER" is polynomially solvable. In order to do so, we start by introducing the following auxiliary problem, which we call FEAS(A):

> **FEAS(A):** Consider an instance of "KMER" and a given set of k-mers $A \subseteq \mathcal{K}$. Does there exist a string t' such that $d_H(s, t') \leq D$ and $K(t') \subseteq A$?

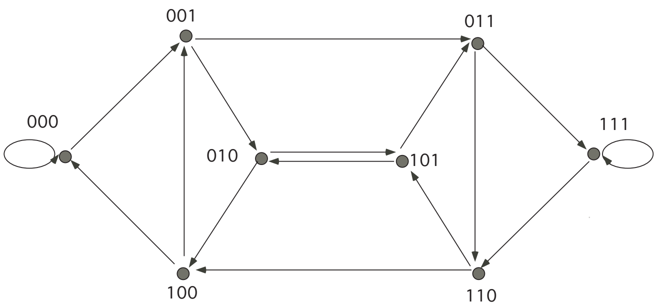

Figure 1. The Shift Register Graph G_3

In the remainder of the section we will prove the following lemma (actually, we obtain a slightly stronger result, i.e., we show how to find t' such that $K(t') \subseteq A$ and $d_H(s,t')$ is minimum):

Lemma 4. *The problem FEAS(A) can be solved in polynomial time.*

We now use Lemma 4 to derive the following theorem.

Theorem 5. *For fixed k and σ, the problem* "KMER" *is polynomially solvable.*

Proof. For fixed k and σ, there are $O(1)$ k-mers, and $O(1)$ possible supports for the solutions (i.e., $O(1)$ possible subsets A of k-mers such that the solution has all its k-mers in A). For each A, enumerated by non-decreasing cardinality, we check in polynomial time (by Lemma 4) if there is a solution t' with support A. We stop as soon as the answer is "yes", and t' is then the optimal solution to "KMER". □

Note that, as $|\mathcal{K}| = \sigma^k$, the above procedure requires to examine, in the worst-case, $O(2^{\sigma^k})$ possible supports. Hence, although polynomial, the algorithm suggested is of little practical use.

We now devote the rest of this section to proving Lemma 4.

The *Shift Register Graph*, also called *De Bruijn Graph*, (SRG, [1]) G_k, for a given k, is a directed graph with node set $\mathcal{K}$, and arcs from i to j whenever $i[2,\cdots,k] = j[1,\cdots,k-1]$ (G_3 is depicted in Figure 1). For $A \subseteq \mathcal{K}$, we denote by $G_k[A]$ the subgraph of G_k induced by the vertices in A. With a slight abuse of notation, we write $(i,j) \in G_k[A]$ to assert that (i,j) is an arc of $G_k[A]$.

We now describe a Dynamic Programming recurrence for the solution of FEAS(A). For $i \in A$ and $h = 1,\ldots,n-k+1$, define $V(i,h)$ to be the minimum Hamming distance between $s[1,\cdots,h+k-1]$ and any string which has all its k-mers in A and ends with k-mer i. We are interested in finding $V(A) := \mathbf{min}_{i\in A} V(i,n-k+1)$. We have the following recurrence, for $1 < h \le n-k+1$ and $i \in A$,

$$V(i,h) = \mathop{\mathbf{min}}_{i' : (i',i)\in G_k[A]} \left(V(i',h-1) + d_H(i[k], s[h+k-1])\right). \tag{1}$$

The boundary conditions are that $V(i,1) = d_H(i, s[1,\cdots,k])$ for all $i \in A$. The complexity of this Dynamic Program is $|A| \times n \times O(\sigma)$, where $O(\sigma)$ is the time of computing the **min** in (1). In fact, given $i \in A$ and a tree T whose leaves are all kmers of A, built as in the proof of Remark 3, each i' such that $(i',i) \in G_k[A]$ can be found as a leaf of T which has the same parent as i. Hence, one only needs to step up one level from i to its parent p, and follow each branch from p to another leaf.

From the above discussion, it follows that the overall time needed to solve "KMER" when σ and k are fixed is $O(2^{\sigma^k}\sigma^{k+1}n)$.

4 NP-hard cases

In this section we show that the problem "KMER" is NP-hard: For completeness, we restate here the problem "KMER":

> **INSTANCE**: An alphabet Σ, an integer k, a string s over Σ, an integer D.
> **PROBLEM**: Find a string $t \in \Sigma^{|s|}$ with $d_H(s,t) \leq D$ and having the smallest possible number of distinct k-mers.

4.1 NP-hardness for fixed *k*

Theorem 6. *The* "KMER" *Problem is NP-hard already for* $k = 2$.

The reduction we propose is from the following problem, called "Compact Bipartite Subgraph":

> **INSTANCE**: A bipartite graph $G = (U, V; E)$, an integer ϕ, an integer λ.
> **PROBLEM**: Find a set of nodes $X \subseteq U \cup V$ with $|X| \leq \phi$ and $|E(G[X])| \geq \lambda$.

Notice that the NP-hardness of the above problem follows trivially from the NP-completeness of "Balanced Complete Bipartite Subgraph". The "Balanced Complete Bipartite Subgraph" problem, also named "GT24" in Garey and Johnson [3], is the following problem.

> **INSTANCE**: A bipartite graph $G = (U, V; E)$, an integer K.
> **QUESTION**: Are there two sets $U' \subseteq U$ and $V' \subseteq V$ with $|U'| = |V'| = K$ and such that $(u, v) \in E(G[U' \cup V'])$ for each $u \in U'$ and $v \in V'$?

And, clearly, the answer to the above question is "yes" if and only if there exists a set of nodes $X \subseteq U \cup V$ such that $|X| \leq 2K$ and $|E(G[X])| \geq K^2$. (For the "only if", notice that if such an X exists then, necessarily, $|X \cap U| = |X \cap V| = K$).

Proof. (Theorem 6) Here we give a reduction from "Compact Bipartite Subgraph" to "KMER". Let A, B be two special symbols and consider $\Sigma = U \cup V \cup \{A, B\}$. Set $D := |E| - \lambda$ and let $M := D + 1$ play the role of a sufficiently big value. Consider the following string $s = s(G, M)$, where the product of two strings denotes their concatenation (and powers are defined accordingly)

$$s = \left(\prod_{v \in V} (Bv)^M \right) B^{2M} \left(\prod_{u \in U} (uB)^M \right) \left(\prod_{(u,v) \in E} (BBuAvBB) \right). \tag{2}$$

A word of explanation is in order to better agree on what the above "simplified" expression for s actually represents: In our intentions, the string s should be considered as uniquely defined. In practice, we refer to implicit orderings of the elements in U, in V, and in E, so that a writing like $\prod_{v \in V}(Bv)$ uniquely defines a string over $V \cup \{B\}$. More precisely, where $v_1, \ldots, v_n$ is an ordering of V, then $\prod_{v \in V}(Bv)$ should be understood as a shorthand for $\prod_{i=1}^{n}(Bv_i)$.

Lemma 7. *There exists a string* $t \in \Sigma^{|s|}$ *with* $d_H(s,t) \leq D$ *and with at most* $1 + 2|U| + 2|V| + \phi$ *distinct* 2*-mers if and only if there exists* $X \subseteq U \cup V$ *with* $|X| \leq \phi$ *and such that* $|E(G[X])| \geq \lambda$.

Proof: Assume given an $X \subseteq U \cup V$ with $|X| \leq \phi$ and such that $|E(G[X])| \geq \lambda$. Consider the following string t, where $S_{(u,v)} := BBuAvBB$ if $(u,v) \in E(G[X])$, and $S_{(u,v)} := BBuBvBB$ if $(u,v) \in E \setminus E(G[X])$:

$$t = \left(\prod_{v \in V} (Bv)^M\right) B^{2M} \left(\prod_{u \in U} (uB)^M\right) \left(\prod_{(u,v) \in E} S_{(u,v)}\right).$$

Notice that $|t| = |s|$ and $d_H(s,t) = |E| - |E(G[X])| \leq |E| - \lambda = D$. Moreover, any 2-mer appearing in t will fall into one of the following categories:

- the single 2-mer BB;
- the $2|U| + 2|V|$ 2-mers of the form zB and Bz for $z \in U \cup V$;
- the $|X \cap U|$ 2-mers of the form uA with $u \in X \cap U$;
- the $|X \cap V|$ 2-mers of the form Av with $v \in X \cap V$.

Hence, the number of distinct 2-mers in t is $1+2|U|+2|V|+|X| \leq 1+2|U|+2|V|+\phi$, as stated.

Conversely, let t be any string such that $|t| = |s|$ and $d_H(s,t) \leq D$. Then the 2-mer BB certainly appears in t since B^{2M}, which is a substring of s, contains M disjoint occurrences of BB. Similarly, for each node $z \in U \cup V$, the 2-mers zB and Bz certainly appear in t. We are assuming that besides these $1 + 2|U| + 2|V|$ 2-mers, string t contains at most ϕ other 2-mers. Let X be made by those $u \in U$ such that the 2-mer uA occurs in t plus the set of those $v \in V$ such that the 2-mer Av occurs in t. Hence, $|X| \leq \phi$. Remember that the string s contains the substring $\prod_{(u,v) \in E}(BBuAvBB)$. Since $d_H(s,t) \leq D$, it follows that $|E \setminus E(G[X])| \leq D$, and hence that $|E(G[X])| \geq |E| - D = \lambda$. □

4.2 NP-hardness for fixed $|\Sigma|$

In this subsection we prove the following.

Theorem 8. *The* "KMER" *problem is NP-hard already for* $|\Sigma| = 2$.

A noticeable property of the proposed reduction is that it constructs instances with $k = O(\log n)$, which allows us to infer the following result.

Theorem 9. *No algorithm solves the* "KMER" *problem in* $O(n^{POLY(k)})$ *time unless* $NP \subseteq DTIME(n^{POLY(\log n)})$. *This negative result holds also for* $|\Sigma| = 2$.

Theorem 9 gives strong evidence that there is no space for improving the $O(n^{|\Sigma|^k})$ approach of Section 3.2. Indeed, an $O(2^{|\Sigma|^k} n^{POLY(|\Sigma|,k)})$ algorithm would imply an $O(n^{POLY(k)})$ algorithm when $|\Sigma| = 2$, and $NP \subseteq DTIME(n^{POLY(\log n)})$ would follow.

The general approach of the reduction is the same as described in Subsection 4.1 In particular, the reduction will be again from "Compact Bipartite Subgraph".

The reduction. As in Subsection 4.1, let $D := |E| - \lambda$ and let $M := D + 1$ play the role of a sufficiently big value. We now work with $\Sigma = \{0, 1\}$. Before explaining how to construct s, t and k, we point out that in the construction given in Subsection 4.1 the fact that certain k-mers had to be present in every string t with $d_H(s,t) \leq D$ played a key role. We hence start the derivation of the reduction to be given here with the consideration that it is relatively easy to build a string s_0 which contains M

disjoint copies of each substring s' in $\{0,1\}^k$ such that either $s'[1] = 1$ or $s'[k] = 1$. Indeed, $s_0 := \left(\prod_{\sigma\in\{0,1\}^{k-1}} 1^k\sigma 1^k\right)^M$ will do the job. Notice also that s_0 contains no k-mer which both starts and ends with 0. Let $h = \lceil \log_2 |U \cup V| \rceil$. Then, to each node $z \in U \cup V$ we can associate a unique binary string $f(z) \in \{0,1\}^h$, called *the short encoding of* z. To each node $z \in U \cup V$ we also associate *the long encoding of* z, denoted by $f'(z) \in \{0,1\}^{2h+1}$, defined as $f'(z)[1] = 0$, $f'(z)[2i] = f(z)[i]$ and $f'(z)[2i+1] = 1$ for each $i = 1, 2, \ldots, h$. When s is a string we denote by $[s]^R$ the reverse of string s. Take $k = 2h + 2$. Consider the following string $s = s(G, M)$

$$s = s_0 \prod_{(u,v)\in E} (1^k\, 0\, f'(u)\, 0\, [f'(v)]^R\, 0\, 1^k).$$

Lemma 10. *There exists a string $t \in \Sigma^{|s|}$ with $d_H(s,t) \le D$ and with at most $3 \cdot 2^{k-2} + \phi$ distinct k-mers if and only if there exists $X \subseteq U \cup V$ with $|X| \le \phi$ and such that $|E(G[X])| \ge \lambda$.*

Proof: The number of strings $s \in \{0,1\}^k$ is 2^k and for 2^{k-2} of them we have that $s[1] = s[k] = 0$. Hence, there are precisely $2^k - 2^{k-2} = 3 \cdot 2^{k-2}$ strings $s \in \{0,1\}^k$ with $s[1] = 1$ or $s[k] = 1$, which account for the numbers occurring in the statement. Assume to be given an $X \subseteq U \cup V$ with $|X| \le \phi$ and such that $|E(G[X])| \ge \lambda$. Consider the following string t,

$$t = s_0 \prod_{(u,v)\in E} S_{(u,v)}\,,$$

where

$$S(u,v) := \begin{cases} 1^k 0 f'(u) 0 [f'(v)]^R\, 0 1^k \text{ if } (u,v) \in E(G[X]) \\ 1^k 0 f'(u) 1 [f'(v)]^R 0 1^k \text{ if } (u,v) \in E \setminus E(G[X]) \end{cases}$$

Notice that $|t| = |s|$ and $d_H(s,t) = |E| - |E(G[X])| \le |E| - \lambda = D$. Moreover, any k-mer appearing in t falls into one of the following categories:

- the $3 \cdot 2^{k-2}$ k-mers starting or ending with a 1 symbol;
- the $|X \cap U|$ k-mers of the form $f'(u)0$ with $u \in X \cap U$;
- the $|X \cap V|$ k-mers of the form $0[f'(v)]^R$ with $v \in X \cap V$.

Hence, the number of distinct k-mers in t is at most $3 \cdot 2^{k-2} + \phi$, as stated.

Conversely, let t be any string such that $|t| = |s|$ and $d_H(s,t) \le D$. Then all the $3 \cdot 2^{k-2}$ k-mers starting or ending with a 1 symbol do certainly appear in t since s contains at least $M > D$ disjoint occurrences of each of them.

Assume that besides these $3 \cdot 2^{k-2}$ k-mers, string t contains at most ϕ other k-mers. Let X be made by those $u \in U$ such that the k-mer $f'(u)0$ occurs in t plus the set of those $v \in V$ such that the k-mer $0[f'(v)]^R$ occurs in t. Hence, $|X| \le \phi$. Remember that the string s contains the substring $\prod_{(u,v)\in E}(1^k 0 f'(u) 0 [f'(v)]^R 0 1^k)$. Since $d_H(s,t) \le D$, it follows that $|E \setminus E(G[X])| \le D$, and hence that $|E(G[X])| \ge |E| - D = \lambda$. □

5 Conclusions

In this paper we have characterized the complexity of the “KMER” problem, under all possible cases for its parameters. For the solution of the NP-hard cases, we have devised Integer Linear Programming formulations, which, for space reasons, are not included in this extended abstract. From our first experimental results, the ILP approach seems suitable for the solution of moderate-size instances of this problem, while for larger-size instances a possibly different (maybe combinatorial) approach should be sought.

References

[1] N. D. de Bruijn: *A combinatorial problem.* Koninklijke Netherlands: Academe Van Wetenschappen, 49 1946, pp. 758–764.

[2] A. Flaxman, A. W. Harrow, and G. B. Sorkin: *Strings with maximally many distinct subsequences and substrings.* The Electronic J. of Combinatorics, 11 2004.

[3] M. R. Garey and D. S. Johnson: *Computers and Intractability, a Guide to the Theory of NP-Completeness*, W.H. Freeman and Co, 1979.

[4] J. D. Watson, M. Gilman, J. Witkowski, and M. Zoller: *Recombinant DNA*, Scientific American Books, W. H. Freeman and Co., 1992.

[5] D. B. West: *Introduction to Graph Theory*, Prentice Hall, 1996.

[6] H. S. Wilf: *Combinatorial Algorithms: An Update*, SIAM CBMS-NSF Regional Conference Series in Applied Mathematics, Society for Industrial and Applied Mathematics, 1989.