International Journal of Foundations of Computer Science

COVERING A CIRCULAR STRING WITH SUBSTRINGS OF FIXED LENGTH

ART M. DUVAL*
Department of Mathematical Sciences, University of Texas at El Paso
El Paso, Texas 79968-0514, U.S.A.

and

W. F. SMYTH†
Department of Computer Science & Systems, McMaster University
Hamilton, Ontario L8S 4K1, Canada
School of Computing, Curtin University
GPO Box U-1987, Perth WA 6001, Australia

Received May 1995
Revised November 1995
Communicated by T. Jiang, Guest Editor

ABSTRACT

A nonempty circular string $C(x)$ of length n is said to be *covered* by a set U_k of strings each of fixed length $k \leq n$ iff every position in $C(x)$ lies within an occurrence of some string $u \in U_k$. In this paper we consider the problem of determining the minimum cardinality of a set U_k which guarantees that *every* circular string $C(x)$ of length $n \geq k$ can be covered. In particular, we show how, for any positive integer m, to choose the elements of U_k so that, for sufficiently large k, $u_k \approx \sigma^{k-m}$, where $u_k = |U_k|$ and σ is the size of the alphabet on which the strings are defined. The problem has application to DNA sequencing by hybridization using oligonucleotide probes.

Keywords: string, cover, algorithm, DNA, sequence, hybridization.

1. Introduction

In the hybridization approach[1] to analysis of a DNA sequence (or string) x of $n = 300 - 500$ base pairs, the base pairs are determined by comparing sections (or substrings) of x of some fixed length k (typically in the range 5–14) with a "chip" U_k which normally contains all possible strings of length k. Thus, in general, for an alphabet Σ of cardinality $\sigma = |\Sigma|$, the number of strings in U_k is σ^k; in particular, for a DNA sequence, $\Sigma = \{C, G, A, T\}$ and $\sigma = 4$. Clearly each such chip, or set of substrings, guarantees that every position in x is contained in a substring which matches one of the substrings on the chip: we say then that U_k *covers* (or is a *cover*

*e-mail: artduval@math.utep.edu
†e-mail: smyth@mcmaster.ca

of) x. Conversely, it is also clear that in order to guarantee that every string x can be covered, the chip U_k must contain *all* strings of length k; otherwise, $y \notin U_k$ would imply that no string $x = yx'$ or $x'y$ could be covered.

It turns out, however, that in practice the "ends" of the string x (the first few characters and the last few characters) can be regarded as being already known and so can be ignored in the covering problem. Thus a combinatorial problem that arises from the hybridization technique is not exactly to cover, but rather to "almost cover" a given string x; that is, for any given string $x = x[1..n]$ of length $n \geq 2k-1$, to design a chip U_k that is guaranteed to contain enough substrings of x to cover the substring $x[k..n-k+1]$ of length $n-2k+2$, even though U_k may not cover x itself. This reformulation of the problem therefore gives rise to the possibility that U_k might not need to contain all possible strings of length k in order to "almost cover" x.

To re-express this problem in a standard combinatorial way, it is convenient to introduce the idea of a "circular" string. Given a string $x = x[1]x[2]\cdots x[n]$ of length n, the *circular string* $y = C(x)$ corresponding to x is defined as follows: for every position $i \in [1..n]$,

$$y[nr+i] = x[i],$$

where r is any integer. Thus $C(x)$ may be thought of as a loop in which $x[1]$ is concatenated at the right of $x[n]$; alternatively, $C(x)$ may be thought of as an infinitely-long string with no start or end position consisting entirely of concatenations of x.

Clearly if U_k covers $C(x)$, then it will also "almost cover" x in the sense described above. Conversely, suppose that U_k almost covers every string x. Observe from the definition of circular string that $C(x)$ is identical to $C(x')$ for any cyclic shift (rotation) x' of x. Since U_k almost covers every such x', it follows that every position in x is covered by some element of U_k, hence that U_k covers $C(x)$. Thus any chip U_k that solves one problem also provides a solution to the other, and so the two problems are indeed equivalent.

It is easy to see that for $\sigma \geq 2$ and $k \geq 2$, every circular string $C(x)$ can be covered by a chip U_k containing fewer than σ^k strings of length k. For example, for $\Sigma = \{a, b\}$, every circular string $C(x)$ on Σ can be covered by the chip $U_2 = \{a^2, ab, b^2\}$ of cardinality $3 < 2^2$. Furthermore, for $k = 2$, there exists no chip of smaller cardinality with this property, as the string $x = a^3b^3ab$ illustrates: in order to cover $C(a^3b^3ab)$, U_k must contain aa, bb, and at least one of ab, ba.

For a given alphabet Σ of cardinality $\sigma \geq 2$ and a given integer $k \geq 2$, these considerations then naturally give rise to the following problems:

(i) Determine the minimum cardinality u_k^* of all chips U_k guaranteed to cover every circular string on Σ.

(ii) Does there exist an efficient algorithm to determine a minimum cardinality chip U_k as specified in (i)?

(iii) If the answer to (ii) is "yes", find such an algorithm; if not, then find an efficient algorithm to compute a chip U_k of "small" cardinality covering every circular string.

In Section 2 of this paper we show how to compute, for any fixed integer m, a chip U_k of asymptotic cardinality σ^{k-m} (though for $m > 1$, k must be much larger than m to achieve this limit), thus providing a partial response to (iii). As explained in Section 2, the approach used in this computation is a highly systematic one, and so raises the possibility that the layout of such chips U_k could be automated. Section 3 deals briefly with some of the consequences of our work in the case $\sigma = 4$ which arises in practice. We note that the notion of cover defined in this paper has some connection to a problem already considered in the literature[5,6]: computing a *single* substring which covers a given linear string x. Related problems on hybridization chips have also been considered.[7,8]

2. Reducing Chip Size by a Polynomial Factor

Throughout this section we consider disjoint nonempty subalphabets Σ_1 and $\Sigma_2 = \Sigma - \Sigma_1$ of the alphabet Σ, of cardinalities σ_1 and σ_2 respectively. For any alphabet Σ and any positive integer m, we shall use the notation Σ^m to denote the set of strings $x = x[1]x[2]\cdots x[m]$ in which each $x[i]$, $1 \leq i \leq m$, is an element of Σ. Thus Σ^m denotes $|\Sigma|^m$ distinct strings. Similarly, the concatenation $\Sigma_1^{m_1}\Sigma_2^{m_2}$ denotes the set of strings xy of length $m_1 + m_2$ with prefix $x \in \Sigma_1^{m_1}$ and suffix $y \in \Sigma_2^{m_2}$. For any fixed integer $k \geq 2$ and any fixed $m = 1, 2, \ldots, k-1$, we construct a chip $U_k^{(m)}$, consisting of all strings u of length k satisfying one of the following two mutually exclusive properties:

- u has a prefix from Σ_1^m;
- u is formed from any combination of the substrings $(\cup_{i=0}^{m-1}\Sigma_1^i)\Sigma_2^1$.

We shall show that each chip $U_k^{(m)}$ covers every circular string $C(x)$ and that $u_k^{(m)} = |U_k^{(m)}|$ is arbitrarily close to $\sigma_1^m\sigma^{k-m}$ for sufficiently large k; that is, $U_k^{(m)}$ is asymptotically smaller by a factor $(\sigma_1/\sigma)^m$ than the regular chip. Then, in particular, by choosing $\sigma_1 = 1$, we can achieve a factor $1/\sigma^m$, as claimed in Section 1. We remark that the same results can also be established for chips $U_k^{(m)}$ containing strings either with *suffix* from Σ_1^m or formed from all combinations of $\Sigma_2^1(\cup_{i=0}^{m-1}\Sigma_1^i)$

Example. We present an example of $U_k^{(m)}$ for $m = 2$ and $\sigma_1 = 1$.

Say $\Sigma = \{A, C, T, G\}$ (as with DNA) and $\Sigma_1 = \{A\}$. Then the two types of strings in $U_k^{(m)}$ are:

- $AA\Sigma^{k-2}$;
- length k combinations of Σ_2 and $A\Sigma_2$.

□

Theorem 1 *$U_k^{(m)}$ covers every nonempty circular string $C(x)$.*

Proof. We show that every position $x[i]$ of $C(x)$ is contained in some element of $U_k^{(m)}$. Consider $y = x[i-k+1..i+m-1]$. If y contains a string from Σ_1^m, say $x[j..j+m-1]$ (so $i-k+1 \leq j \leq i$), then $x[j..j+k-1]$ is a string of length k with prefix from Σ_1^m containing $x[i]$. Otherwise $x[i..i+m-1]$ contains a character from Σ_2, say $x[j]$ (so $i \leq j \leq i+m-1 \leq i+k-1$), and then $x[j-k+1..j]$ is a string of length k with no substring from Σ_1^m, ending in a character from Σ_2, and containing $x[i]$. $\square$

Observe that Theorem 2.1 holds even when $n = |x| < k$.

To determine the cardinality of $U_k^{(m)}$, we begin with the special case $m = 1$. Then $U_k^{(1)}$ consists of all strings $\Sigma_1^1 \Sigma^{k-1}$ together with all strings Σ_2^k. Hence

$$u_k^{(1)} = \sigma_1 \sigma^{k-1} + \sigma_2^k,$$

an expression which achieves its minimum when $\frac{du_k^{(1)}}{d\sigma_1} = 0$; that is, when

$$\sigma_2 = \sigma / \sqrt[k-1]{k}. \tag{1}$$

Since the denominator in this expression approaches unity as $k \to \infty$, it follows that for all sufficiently large k, $\sigma_2 = \sigma - 1$ and $\sigma_1 = 1$, so that

$$\frac{u_k^{(1)}}{\sigma^k} = \frac{1}{\sigma} + \left(\frac{\sigma-1}{\sigma}\right)^k. \tag{2}$$

Thus, asymptotically, $U_k^{(1)}$ requires only $\frac{1}{\sigma}^{\text{th}}$ of the number of strings stored in the complete chip. In the case $\sigma = 4$ which arises in DNA sequencing, the range 5..14 for k yields values of $\sqrt[k-1]{k}$ in (1) ranging from $\sqrt[4]{5} = 1.495$ to $\sqrt[13]{14} = 1.225$, so that for each of these values of k, $\sigma_2 = \sigma - 1$. Over the same range of k, $\left(\frac{\sigma-1}{\sigma}\right)^k$ reduces from 0.237 to 0.018, so that for large chips the saving is close to the asymptotic lower bound of 1/4 implied by (2). For further details related to the case $\sigma = 4$, see Section 3.

In order to give an explicit formula for the case $m \geq 2$, recall first the definition of the multinomial coefficient[4]:

$$\begin{aligned}\binom{k_1+k_2+\cdots+k_m}{k_1,k_2,\ldots,k_m} &= \frac{(k_1+k_2+\cdots+k_m)!}{k_1!k_2!\ldots k_m!} \\ &= \binom{k_1+k_2}{k_1}\binom{k_1+k_2+k_3}{k_1+k_2}\cdots\binom{k_1+k_2+\cdots+k_m}{k_1+\cdots+k_{m-1}},\end{aligned}$$

valid for integers $k_i \geq 0$, $i = 1, 2, \ldots, m$. Thus for $m = 2$, the multinomial coefficient reduces to the ordinary binomial coefficient. The number of elements in $U_k^{(m)}$ may now be counted as follows:

Theorem 2 *For every integer $m \in 2..k-1$,*

$$u_k^{(m)} = \sigma_1^m \sigma^{k-m} + \sigma_2^k \sum_{S_m = k} \binom{k_1+k_2+\cdots+k_m}{k_1,k_2,\ldots,k_m} \left(\frac{\sigma_1}{\sigma_2}\right)^{k_2+2k_3+\cdots+(m-1)k_m}, \tag{3}$$

where $S_m \equiv k_1 + 2k_2 + \cdots + mk_m$.

Proof. The term $\sigma_1^m \sigma^{k-m}$ just counts the number of strings in $U_k^{(m)}$ with prefix from Σ_1^m, so we need to show that the summation term in (3) counts the strings formed from $T_m = (\cup_{i=0}^{m-1} \Sigma_1^i)\Sigma_2^1$. Denoting by k_i the number of occurrences of strings from $\Sigma_1^{i-1}\Sigma_2$, $1 \leq i \leq m$, in any combination formed from T_m, we see that every such combination must satisfy $S_m = k$, since the total length of the string formed must always be k. For each valid choice of the k_i, the number of distinct combinations is just[3]

$$\binom{k_1+k_2+\cdots+k_m}{k_1,k_2,\ldots,k_m}.$$

The terms in each of these combinations can be chosen in exactly

$$\sigma_2^{k_1}(\sigma_1\sigma_2)^{k_2}\ldots(\sigma_1^{m-1}\sigma_2)^{k_m}$$

distinct ways. Putting together these observations and doing some routine simplification yields (3). □

The expression in (3) is used in Section 3 to compute minimum chip sizes for particular values of k. We use a somewhat different approach to determine the asymptotic behavior of $u_k^{(m)}$ in our main result:

Theorem 3 *For every integer $m \geq 1$, $\lim_{k\to\infty} u_k^{(m)}/\sigma^k = (\sigma_1/\sigma)^m$.*

Proof. Fix $m \leq k$. Define T_m as in the proof of Theorem 2, and let $a_k^{(m)}$ be the number of strings of length k formed from T_m. Since $u_k^{(m)} = \sigma_1^m\sigma^{k-m} + a_k^{(m)}$ (as in Theorem 2), it will suffice to show that $\lim_{k\to\infty} a_k^{(m)}/\sigma^k = 0$.

Let y be a string of length k formed from T_m. Write $y = y_1y_2\cdots y_py_{p+1}$ where each y_i $(i = 1..p)$ has length m, and y_{p+1} has length $r < m$ (so $k = mp + r$). Now y does not contain m consecutive characters from Σ_1, so $y_i \notin \Sigma_1^m$ $(i = 1..p)$, and thus there are at most $\sigma^m - \sigma_1^m$ possibilities for each y_i $(i = 1..p)$. Of course, there are at most σ^r possibilities for y_{p+1}. Therefore $a_k^{(m)} \leq (\sigma^m - \sigma_1^m)^p\sigma^r = (\sigma^m - \sigma_1^m)^{(k-r)/m}\sigma^r$.

It follows that

$$\frac{a_k^{(m)}}{\sigma^k} \leq \frac{(\sigma^m-\sigma_1^m)^{(k-r)/m}}{\sigma^{k-r}} = \left(\left(\frac{\sigma^m-\sigma_1^m}{\sigma^m}\right)^{1/m}\right)^{k-r} < \left(\left(\frac{\sigma^m-\sigma_1^m}{\sigma^m}\right)^{1/m}\right)^{k-m},$$

which tends to 0 as $k \to \infty$ (for fixed m), since $((\sigma^m - \sigma_1^m)/\sigma^m)^{1/m} < 1$. □

Theorem 3 only proves that $u_k^{(m)}/\sigma^k$ converges to $(\sigma_1/\sigma)^m$. The rate of convergence, however, depends essentially on how fast b^k converges to 0, where $b = (1 - (\sigma_1/\sigma)^m)^{1/m}$. As m gets larger, b gets closer to 1 very quickly, and the rate of convergence gets slower. So, although Theorem 3 seems to promise an arbitrary reduction in chip size (by a factor of $(\sigma_1/\sigma)^m$ for any m), to actually achieve this reduction requires k to be sufficiently large compared to m. In the case $\sigma = 4$ of DNA sequencing, $m = 1$ yields the smallest chip for every $k \leq 18$. See Section 3 for more details.

3. Practical Considerations

We present here the results of computer experiments for $4 \leq k \leq 85$, in which a minimum chip size for the alphabet $\sigma = 4$ that arises in practice is determined. For every value of k, $u_k^{(m)}$ was computed for every possible value of m (thus, for $1 \leq m \leq k-1$) and for every possible choice of σ_1 (thus, for $1 \leq \sigma_1 \leq \sigma - 1 = 3$), using Theorem 2. Then, for each k, the minimum of these values $u_k^{(m)}$ was used to compute the minimum ratio $u_k^{(m)}/4^k$. The results are shown in Table 1 with the corresponding values of m and σ that yielded the minimum ratio. It is a little surprising that the case $\{m = 3, \sigma_1 = 2\}$ intervenes for $19 \le k \le 44$ before $\{m = 2, \sigma_1 = 1\}$ takes over for $45 \le k \le 85$; the value $u_{85}^{(2)}/4^{85} = 0.0712$ is close to the minimum value 0.0625 for $m = 2$, but the capabilities of the computer did not permit further cases beyond $k = 85$ to be investigated. Apart from Theorem 3, nothing is known about the asymptotic behavior of these cases. Of course the values of k considered here are well beyond the upper bounds achievable by current hybridization technology.

Table 1. Minimum Ratio for $\sigma = 4$, $4 \le k \le 85$

k	m	σ_1	$u_k^{(m)}/4^k$
4	1	2	0.5625
5–18	1	1	0.4873–0.2556
19–44	3	2	0.2509–0.1405
45–85	2	1	0.1367–0.0712

Finally, we remark that the minimum chip problem can be modeled as a problem in graph theory.[2] This model may provide clues as to whether or not the problem is indeed NP-hard.

Acknowledgements

The work of the first author was supported in part by Grant No. 1G12 RR08124-01 of the National Institutes of Health through the BioStatistical Laboratory at the University of Texas at El Paso. The work of the second author was supported in part by Grant No. A8180 of the Natural Sciences & Engineering Research Council of Canada and by Grant No. GO-12778 of the Medical Research Council of Canada. The authors express their gratitude to Rade Drmanac for introducing them to the problem, and to the referees for several valuable improvements, especially in the proofs of Theorems 1 and 3.

References

1. R. Drmanac and R. Crkvenjakov, "Sequencing by hybridization (SBH) with oligonucleotide probes as an integral approach for the analysis of complete genomes," *Int. J. of Genome Research* **1-1** (1992) 59–79.
2. A. M. Duval and W. F. Smyth, "Reduced correlation graphs," work in progress (1995).
3. W. Feller, *Probability Theory & Its Applications*, vol. 1, (John Wiley & Sons, New York, 1950), p. 32.
4. D. E. Knuth, *The Art of Computer Programming*, vol. 1, 2nd ed., (Addison-Wesley, Reading, Mass., 1973), p. 64.
5. D. Moore and W. F. Smyth, "An optimal algorithm to compute all the covers of a string," *IPL* **50** (1994) 239–246.
6. D. Moore and W. F. Smyth, "Correction to: An optimal algorithm to compute all the covers of a string," *IPL* **54** (1995) 101–103.

7. P. A. Pevzner and R. J. Lipshutz, "Toward DNA sequencing by hybridization," preprint (1993).
8. S. S. Skiena and G. Sundaram, "Reconstructing strings from substrings," preprint (1993).