Reconstructing Strings from Substrings in Rounds

Dimitris Margaritis
Department of Computer Science
State University of New York
Stony Brook, NY 11794-4400
dmarg@sbcs.sunysb.edu

Steven S. Skiena *
Department of Computer Science
State University of New York
Stony Brook, NY 11794-4400
skiena@sbcs.sunysb.edu

Abstract

We establish a variety of combinatorial bounds on the tradeoffs inherent in reconstructing strings using few rounds of a given number of substring queries per round. These results lead us to propose a new approach to sequencing by hybridization (SBH), which uses interaction to dramatically reduce the number of oligonucleotides used for de novo sequencing of large DNA fragments, while preserving the parallelism which is the primary advantage of SBH.

1 Introduction

Sequencing by hybridization (SBH) [4, 11] is a new and promising approach to DNA sequencing which offers the potential of reduced cost and higher throughput over traditional gel-based approaches. In this paper, we propose a new approach to sequencing by hybridization which permits the sequencing of arbitrarily large fragments without the inherently exponential chip area of SBH, while retaining the massive parallelism which is the primary advantage of the technique. We establish the potential of our technique through both analytical results and simulation on real DNA sequences. Our approach is based on our solution of an interesting combinatorial problem, that of reconstructing strings from substrings in rounds.

The traditional sequencing by hybridization procedure attaches a set of single-stranded fragments to a substrate, forming a *sequencing chip.* A solution of radiolabeled single-stranded target DNA fragments are exposed to the chip. These fragments hybridize with complementary fragments on the chip, and the hybridized fragments can be identified using a nuclear detector. Each hybridization (or the lack thereof) determines whether the string represented by the fragment is or is not a substring of the target. The target DNA can now be sequenced based on the constraints of which strings are and are not substrings of the target. Pevzner and Lipshutz [11] give an excellent survey of the current state of the art in sequencing by hybridization, both technologically and algorithmically.

The most widely used sequencing chip design, the classical sequencing chip $C(m)$, contains all 4^m single-stranded oligonucleotides of length m. For example, in $C(8)$ all $4^8 = 65,536$ octamers are used. Pevzner's algorithm [12] for reconstruction using classical sequencing chips interprets the results of a sequencing experiment as a subgraph of the de Bruijn graph, such that any Eulerian path corresponds to a possible sequence. Thus the reconstruction is not unique unless the subgraph consists entirely of a directed induced path.

Fragment	Classical SBH		Interactive SBH	
Length	Length	Size	Rounds	Size
80	7	16,384	7	560
180	8	65,536	8	1,440
260	9	262,144	8	2,080
560	10	1,048,576	8	4,480
1300	11	4,194,304	9	11,700
2450	12	16,777,216	9	22,050

Table 1: Characteristic length of unambiguously deciphered DNA fragment as a function of the size for classical and interactive SBH.

The strength of this requirement means that enormous sequencing chips are needed to reconstruct relatively short strands of DNA. For example, the classical chip $C(8)$ suffices to reconstruct 200 nucleotide long sequences in only 94 of 100 cases [10], even in error-free experiments. Unfortunately, as shown in Table 1, the length of unambiguously reconstructible sequence grows slower than the size (ie. area) of the chip. Thus exponential growth inherently limits the length of the longest reconstructible sequence by traditional SBH.

Our approach uses *interaction* to reduce the required amount of work. Suppose that we are given an unknown string S, over a known alphabet Σ, and are permitted to ask questions of the form "is s a substring of S?", where s is a specific query string over Σ. We are not told where s occurs in S, nor how many times it occurs, just whether or not s a substring of S. We need to be able to ask many useful questions simultaneously in order to minimize set-up costs and elapsed time. Therefore, our goal is to determine the exact contents of S using as few rounds of as few queries as possible. The results of Table 1 demonstrate that we succeed, using less than ten rounds to reduce the total number of queries by a several orders of magnitudes on reasonable sized fragments.

*Corresponding author. This work was partially supported by ONR award 400x116yip01.

Other variants of SBH (such as nested-strand SBH [13, 14] and positional SBH [2, 6]) have been proposed to increase the resolving power of classical SBH. However, none of them offer the potential of interactive SBH to sequence very long fragments.

Skiena and Sundaram [15] studied the complexity of *sequentially* reconstructing unknown strings from substring queries. Specifically, they show that $(\alpha - 1)n + \Theta(\alpha\sqrt{n})$ queries are sufficient to reconstruct an unknown string, where α is the alphabet size and n the length of the string, matching the information-theoretic lower bound for binary strings. Further, they show that $\sim \alpha n/4$ queries are necessary, which is within a factor of 4 of the upper bound for larger alphabets. However, achieving a high degree of parallelism is critical for this approach to lead to a practical method of DNA sequencing.

In this paper:

- We show a wide range of tradeoffs between the number of rounds of substring queries and the number of queries per round sufficient to determine an unknown string of length n on an alphabet of size α. Our results are summarized in the table below:

Number of Rounds	Questions per Round
αn	1
n	α
$\lg^2 n$	n
$\lg n$	$n^2/\lg n$
$\lg\lg n$	$\alpha^{\alpha(1+o(1))(\lg n/\lg\lg n)}$
2	$\alpha^{O(\sqrt{n\lg n})}$
1	$3\alpha^{\lfloor n/2\rfloor+1}$

  Each of these tradeoffs require different ideas to achieve.

- We prove an exponential lower bound on the capacity of any prefabricated sequencing chip capable of reconstructing n-strings, and give a new chip design whose capacity approaches this lower bound.

- We give a strategy which uses (with probability $1-1/n^{\epsilon}$) $O(\alpha\epsilon\lg n)$ rounds of n queries per round, and present simulation results which demonstrate the practicality of this approach. Indeed, our simulations suggest a much stronger result, that far fewer number of rounds of n queries per round suffice to reconstruct sequences of length n. This suggests a very efficient technique to sequence large DNA fragments, and also an application to designing custom, prefabricated chips to identify mutations for diagnostic purposes.

In the sense of being biological techniques proposed by computer scientists, our work is philosophically akin to the recent work of Adleman [1] and Lipton [8] on biocomputing. We stress that for our proposed techniques, the issue is not one of feasibility, but of cost, since either the photolithography methods of [4] or the primer walking technique of [7] can be used to realize interactive SBH, albeit in an expensive manner. We are confident that biologists can develop experimental protocols to reduce these costs, once they understand the combinatorial advantages of our approach.

In Section 2, we analyze the one round case, corresponding to conventional sequencing chips, and establish tight upper and lower bounds for capacity. In Section 3, we establish bounds on chip capacity sufficient to achieve a sub-logarithmic number of rounds. In Section 4, we present our most interesting tradeoffs, showing that polylog rounds of subquadratic capacity chips always suffice for reconstruction. Finally, in Section 5 we prove better bounds in the average case. In an appendix, we give the results of simulations to establish the practicality of our techniques.

2 Reconstruction in One Round

In this section, we consider the problem of reconstructing strings using a fixed set of queries, as in conventional SBH. A *sequencing chip* C is defined by a given set of query strings $c_1, \ldots, c_m$ over a given alphabet Σ. The *capacity* or *size* m of the chip is the number of strings which define it. The *spectrum* $Sp(C, S)$ of chip C with respect to string S partitions the strings of C into two sets, those which are substrings of S and those which are not. A string S can be *reconstructed* with a given chip C iff there does not exist a string $S' \in \Sigma^*$ such that $Sp(C,S) = Sp(C,S')$. In other words, the spectrum of S uniquely describes S.

In this section, we consider the question of minimizing the size of any chip capable of reconstructing all strings of length n. Clearly, a chip containing all α^n strings of length n suffices for reconstruction, since the spectrum of any string S will contain only one positive substring, ie. S itself. However, significantly smaller chips are in fact possible.

Consider a classical sequencing chip $C(l)$, where $l = \lfloor n/2\rfloor + 1$, consisting of all α^l l-strings. A string S has *period* k if $S_i = S_{i+k}$ for all $1 \leq i \leq n-k$. Observe that strings of period $k \leq l$ cannot be reconstructed using $C(l)$. For example, the strings *abcdabc*, *bcdabcd*, *cdabcda*, and *dabcdab* all contain exactly the same set of 4-substrings; *abcd*, *bcda*, *cdab*, and *dabc*. Thus $C(l)$ does not suffice for reconstructing n-strings, but we shall show that a slightly larger chip does while no smaller chip can.

Our arguments will be based on two n-strings S and T, both composed of the same set of l-strings. Let $S[i]$ denote the ith character of S. Let S_i (T_i) denote the l-string beginning in the ith position of S (T), ie. the ith through $(i+l-1)$th characters. Thus S_i and S_{i+1} share $l-1$ characters in common. Given S, string T is completely described by a permutation P_T of the $(n-l+1)$ l-substrings of S.

Lemma 1 *Let S and T be distinct n-strings such that $Sp(C(m), S) = Sp(C(m), T)$, where $C(m)$ is the classical sequencing chip and $m > n/2$. If $T_x = S_{j+1}$ and $T_{x+p} = S_i$, then $S[z] = S[z + (j-i+1+p)]$ for $1 \leq z \leq m-p$. In other words, the first $m+j-i+1$ characters of S form a period $(j-i+1+p)$ string.*

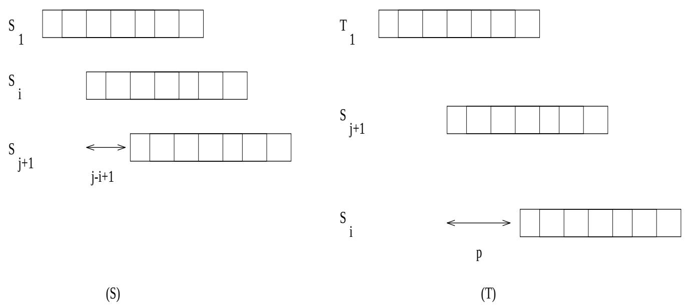

Figure 1: Forcing a low-period substring.

Proof: Figure 1 illustrates the situation. By definition, $S_i[z+j-i+1] = S_{j+1}[z]$ for $1 \leq z \leq m-(j-i+1)$. Similarly, $T_{x+p}[z] = T_x[z+p]$ for $1 \leq z \leq m-p$. Therefore, $S_i[z] = S_{j+1}[z+p]$ for $1 \leq z \leq m-p$, and the result follows since $S_x[y] = S[x+y-1]$. ∎

We say a permutation P on $\{1,\ldots,n\}$ is *prefix-separable* if there exists $k < n$ such that $\max(P_1,\ldots,P_k) = k$.

Lemma 2 *Let S and T be distinct n-strings such that $Sp(C(m),S) = Sp(C(m),T)$ and P_T is prefix-separable. Then there exist proper distinct n'-strings S' and T' such that S' is a substring of S and T' is a substring of T, where $Sp(C(m),S') = Sp(C(m),T')$.*

Proof: Since P_T is prefix-separable, the sets of the first k m-strings of S and T are identical, as are the sets of the last $(n-k-m)$ m-strings. If either $S[1,m+k-1] \neq T[1,m+k-1]$ or $S[k+1,n] \neq T[k+1,n]$, we have our S' and T'. If not, $S=T$ and S and T were not distinct n-strings. ∎

Lemma 3 *Let S and T be distinct n-strings such that $Sp(C(m),S) = Sp(C(m),T)$, where $C(m)$ is the classical sequencing chip, P_T is not prefix-separable, and $m > n/2$. Then for any $i < n-m$, there exists a $j > i$, p, and x such that $(j-i+1+p) \leq m$, $T_x = S_{j+1}$ and $T_{x+p} = S_i$. Thus, $S[i, m+j+1]$ is a period $(j-i+1+p) \leq m$ string.*

Proof: Since P_T is not prefix-separable, for any $i < n$, there exists at least one $j > i$ such that the starting position of S_{j+1} is to the left of S_i in T. Let j be the smallest such integer. Thus $S_{i+1},\ldots,S_j$ in T must all be to the right of the starting position of S_i in T. Thus there can be *at most* $m-1-(j-i)$ starting positions to the left of S_i in T, and $p \leq m-j+i-1$.

Therefore, by Lemma 1, $S[i, m-j+1]$ is a period $(j-i+1+p)$ string. Since $j-i+1+p \leq j-i+1+(m-j+i-1) = m$, this string has period at most m. ∎

By the GCD Lemma of Lyndon and Schutzenberger [9]:

Lemma 4 *Let $S = abc$ be a string formed by concatenating strings a, b, and c. Let string ab have period i, and string bc have period j, where $|b| > \max(i,j)$. Then*

- *If $i = j$, then S has period $i = j$.*
- *If $i = d \cdot j$, for integer d, S has period i.*
- *If $i \perp j$, S has period 1.*

Lemma 5 *The classical chip $C(m)$ suffices to reconstruct any n-string of period $k > m$ if $m > n/2$.*

Proof: We show that if $Sp(C(m),S) = Sp(C(m),T)$, and $S \neq T$, then both S and T must have period at most m. Consider the smallest sized counter-example. Assume P_T is prefix-separable. By Lemma 3, for all $1 \leq i \leq n-m$, the substring $S_i = S[i, m+j+1]$ has period $\leq m$ for some $j \geq i$. Thus S_i overlaps S_{i+1} in at least m positions. By the GCD Lemma of [9], the intersection of these strings must have period at most m, which can be extended to comprise all of S. The case of T follows by symmetry.

Now suppose P_T is not prefix-separable. By Lemma 2 this could not be the smallest counter-example, a contraction and the result follows. ∎

Lemmas 5 leads directly to an efficient sequencing chip design:

Theorem 6 *One round of $3\alpha^{\lfloor n/2 \rfloor+1}$ queries suffices to reconstruct any n-string on an alphabet Σ, $\alpha = |\Sigma|$.*

Proof: Our sequencing chip consists of all distinct $(\lfloor n/2 \rfloor + 1)$-strings, plus all n-strings of period at most $(\lfloor n/2 \rfloor + 1)$. By Lemma 5, the former are sufficient to reconstruct any long period strings, and none of the latter strings will prove substrings of the unknown long-period string. If the unknown string has a short period, exactly one of the latter strings will prove a substring. This chip contains $\alpha^{\lfloor n/2 \rfloor+1} + \sum_{i=1}^{\lfloor n/2 \rfloor+1} \alpha^i \leq 3\alpha^{\lfloor n/2 \rfloor+1}$ strings. ∎

Period $\lfloor n/2 \rfloor + 1$ strings prove the key to the lower bound as well:

Theorem 7 *Any sequencing chip capable of reconstructing all strings of length n must have size at least $2\alpha^{\lfloor n/2 \rfloor+1}/n - 1$.*

Proof: Let $l = \lfloor n/2 \rfloor + 1$. Consider a set of questions adequate to reconstruct each of the α^l the period-l n-strings. We can partition these strings into equivalence classes, where each of these n-strings are equivalent under circular shifts. Thus the size of these equivalence classes ranges from one to l, depending upon the minimum period of the string. Observe that:

- Any question capable of distinguishing between members of the same equivalence class must have length greater than l, since all the strings in the same class have the same l-spectrum.

- *Any* question of length greater than l will be substrings of members of at most one equivalence class, since all l-substrings of this question must be equivalent under circular shift (if not, the question will be a substring of none of the period l strings).

Thus none of the $\lg |C| \geq 1$ queries necessary to distinguish between the C members of an equivalent class can possibly be substrings of any string outside the equivalence class. Since each equivalence class contains at most l members, there are at least α^l/l such classes, and the result follows from making at least one query per class. ∎

3 Reconstruction using Few Rounds

To show how interaction can be used to reduce the total number of queries, we first review the results of Skiena and Sundaram [15] on reconstructing strings with one substring query per round. A subtlety of the problem is whether the length of the unknown string is presented in advance, or must be determined using the results of queries. For ease of exposition, we will assume that the length n is known, since it results in simpler strategies whose complexities are identical except for lower order terms.

Lemma 8 *An unknown string S of known length n on alphabet Σ, $|\Sigma| = \alpha$ can be reconstructed in $\alpha(n+1)$ substring queries.*

Proof: Begin by making substring queries of single-character substrings, so after at most α queries we know a character of S. Let s be a known substring of S and $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_\alpha\}$. In general, we can increase the length of this known substring by one character by querying on the strings $s\sigma_i$, for $1 \leq i \leq \alpha$. At least one of these query strings must be a substring of S, unless s is a suffix of S. When s can no longer be extended, s is a suffix of S and we can continue the process by prepending each character to the known substring, until it is of length n and S is determined. ∎

In [15], we show how to reduce the multiplicative constant by one, ie. that $(\alpha-1)n + 2\lg n + O(\alpha)$ substring queries suffice if n is known, while if n is unknown, $(\alpha-1)n + \Theta(\alpha\sqrt{n})$ queries suffice. With our lower bounds of $(\lg \alpha)n$ and $\alpha n/4$, this strategy is tight for binary strings and within a factor of 4 of optimal over any alphabet.

The strategy of Lemma 8 can be parallelized in a trivial way, by observing that each of the α extension queries can be done in parallel, yielding:

Corollary 1 *$n+1$ rounds of α substring queries per round suffice to reconstruct an unknown string of length n on an alphabet of size α.*

In the rest of this section, we use a divide-and-conquer approach to deliver a much higher degree of parallelism.

Lemma 9 *Any string S on an alphabet of size α can be reconstructed using at most r rounds of $\alpha^{n^{1/r}\lg^{(r-1)/r} n}$ substring queries per round.*

Proof: Consider the following r round reconstruction strategy, which is parameterized by the constants $k_1, \ldots, k_r$:

- *round 1*: Query all α^{n/k_1} strings of length n/k_1. Let S_1 denote the resulting set of substrings of S of length n/k_1.

- *round* $2 \leq i \leq r$: Let S_{i-1} denote the set of all of the (at most n) distinct $(\prod_{j=2}^{i-1} k_j \cdot (n/k_1))$-substrings of S. Query all of the n^{k_i} strings which can be formed as a sequence of k_i elements of S_{i-1}.

This strategy is correct whenever $\prod_{j=2}^{r} k_j/k_1 \geq 1$, as S_i is determined at the end of round i, and $S_r = S$. We select k_1 and k_j $(2 \leq j \leq r)$ to satisfy the following relations:

$$n/k_1 - \lg n \cdot k_j$$

$$k_1 = k_j^{r-1}$$

Solving for k_1 and k_j yields:

$$k_1 = (n/\lg n)^{(r-1)/r}$$

$$k_j = (n/\lg n)^{1/r}$$

In the first round, $\alpha^{n/k_1} = \alpha^{n^{1/r}\lg^{(r-1)/r} n}$ queries are made. In the second through rth rounds, $n^{k_j} = \alpha^{n^{1/r}\lg^{(r-1)/r} n}$ queries are made, giving the result. ∎

Corollary 2 *Any string of length n on an alphabet of size α can be determined using 2 rounds of $\alpha^{\Theta(\sqrt{n \lg n})}$ queries per round.*

Corollary 3 *Any string of length n on an alphabet of size α can be determined using $\lg\lg n$ rounds of $\alpha^{\alpha(1+o(1))(\lg n/\lg\lg n)}$ queries per round.*

Corollary 4 *Any string of length n on an alphabet of size α can be determined using $\lg n$ rounds of n^α queries per round.*

4 Reconstruction using Polylog Rounds

The results in the previous sections demonstrate that it is possible to reconstruct strings from substrings in few rounds, but at a cost of an exponential number of queries per round. Practical implementation of interactive SBH forbids such extravagance –

the largest currently realized sequencing chip contains only 65,384 oligonucleotides. We seek to reconstruct long sequences with chips of capacity on this order of magnitude.

In this section, we consider strategies which use a polylogarithmic number of rounds, but a low-order polynomial number of queries per round. Our algorithms are based on the following observations:

Lemma 10 *A string S of length n contains $\leq n-l+1$ different substrings of length l.*

Proof: Any particular l-substring has a unique first starting position in S, since no two distinct l-strings can begin at the same position in S. The result follows since there are only $n-l+1$ possible starting positions in S. ∎

Lemma 11 *Given the set of all distinct l-substrings of S, $|S| = n$, one round of $(n-l+1)^2$ queries suffice to find all distinct $2l$-substrings of S.*

Proof: Any $2l$-substring of S can be formed by concatenating two l-substrings of S. By Lemma 10, there are only a linear number of l-substrings. ∎

Lemma 11 immediately gives an algorithm for reconstructing strings in $\lceil \lg n \rceil$ rounds of n^2 queries, by starting with one character queries and repeatedly doubling. This strategy may be seen as wasteful, however, since some of the n^2 concatenations may contain l-strings which are not l-substrings of S. These prospective queries can be eliminated without effecting the accuracy of the algorithm.

Theorem 12 *$O(\lg n)$ rounds of $n^2/\lg n$ substring queries per round suffice to reconstruct any string of length n on an alphabet of size $\alpha \leq n$.*

Proof: We use the previously described doubling strategy, where we ask queries concatenating two l-substrings iff all l distinct l-substrings of the length $2l$ queries are in fact substrings of S. Thus the algorithm proceeds in $\lg n$ meta-rounds, where the ith meta-round consists of m_i queries surviving from at most $n - 2^i - 1$ candidates. If we are restricted to rounds of $n^2/\lg n$ queries, the total number of rounds in this strategy is given by

$$R = \sum_{i=1}^{\lg n} \lceil m_i/(n^2/\lg n) \rceil$$

We analyze the complexity of R by partitioning all queries asked into two sets, those queries which prove to be substrings of S (ie. return 'true' to the query) and those queries which prove not to be substrings of S. By Lemma 10, at most n queries per round can be substrings of S, for a total of at most $n \log n$ queries in the first set.

A no-query asked in round i corresponds to the concatenation xy of two 2^{i-1}-substrings of S, where x ends at position $p(x)$ in S, y begins at position $p(y)$ in S, and $p(x) \neq p(y)$. In no subsequent round, will a query be asked concatenating a string ending in $p(x)$ with a string beginning in $p(y)$, because such a query will contain xy, which is known not to be a substring of S. Thus at most n^2 queries will prove to be no-queries, and

$$\sum_{i=1}^{\lg n} m_i = n^2 + n \log n$$

Subject to this contraint, R is maximized at $2 \lg n$, giving the result. ∎

Theorem 12 gets us to a tradeoff approaching practicality, but $n^2/\log n$ queries per round still appears too large to sequence long pieces of DNA. For $n > 1000$, we exceed the capacity of the largest sequencing chip constructed to date. Below, we consider efficient strategies using a linear number of queries per round.

Lemma 13 *Consider a set U of m strings on alphabet Σ, $|\Sigma| = \alpha$, where each string begins with the same substring s. There exists a string s' which is contained in at least $m/(2\alpha+1)$ and at most $2m\alpha/(2\alpha+1)$ strings of U.*

Proof: By definition, s is a prefix of each string in U. We will construct s' one character at a time, extending it by the character $c \in \Sigma$ such that cs' or $s'c$ is a substring of the largest number of strings in U. Since there are only 2α possibilities, by the pigeonhole principle the most popular character/position pair will reduce the cardinality of U by a factor of at most $1/2\alpha$. We stop extending s' when the number of remaining strings lies between $m/(2\alpha+1)$ and $2m\alpha/(2\alpha+1)$, as it must eventually do. ∎

Lemma 14 *Given the set of all distinct l-substrings of S, $|S| = n$, $O(\log_{(1+\alpha/\alpha)} n)$ rounds of n queries suffice to find all distinct $2l$-substrings of S.*

Proof: We construct the set of $\leq n^2$ concatenation strings xy, and distribute them into $\leq n$ piles, where pile $p(x)$ consists of all concatenation strings sharing same the l-substring x.

For each pile, we use Lemma 13 to identify a string q_1 which which partitions the pile into two smaller but roughly equal-sized piles, $p_{1y}(x)$ containing q_1 and $p_{1n}(x)$ not containing q_1.

Applying Lemma 13 to each of these piles yields a total of two more query strings (q_2 for $p_{1y}(x)$ and q_3 for $p_{1n}(x)$) which partitions $p(x)$ into four roughly equal-size piles. There are eight possible outcomes to the set of queries q_1, q_2, and q_3. If q_1 returns false, all of the candidates in pile $p_{1y}(x)$ can be eliminated, as all of these contain q_1 where S does not. This test is not symmetrical, however. If q_1 returns true, we cannot eliminate the candidates of $p_{1n}(x)$, because all we have proven is that S must contain q_1 somewhere but this does not preclude it from containing substrings in pile $p_{1n}(x)$.

If either of queries q_2 or q_3 return false, all the candidates in at least one subpile can be eliminated, reducing the size of the original pile by a constant fraction. All three queries return true only if there exist at least two distinct substrings in S beginning with x, with one in $p_{1y}(x)$ and another in $p_{1n}(x)$.

Thus we have shown that after three queries per pile, each pile is either reduced by a constant fraction or split into roughly equal subpiles. Each subpile is defined by a substring starting from a unique position in S, so there can never be more than n active subpiles. Thus in $O(\lg n)$ rounds of n queries per round, each pile can be can be reduced to at most one string per pile, each corresponding to a distinct $2l$-substring of S. Further, each of the $2l$-substrings must represented by a pile if the given set of l-substrings was indeed complete. ▮

Performing the $\lg n$ meta-round doubling strategy of Theorem 12 with the pruning implementation of Lemma 14 gives:

Theorem 15 $O(\lg n \cdot \log_{(1+\alpha)/\alpha} n)$ *rounds of* n *substring queries per round suffice to reconstruct a string* S *of length* n

5 Probabalistic Analysis

Throughout this paper we have been concerned with worst-case results. In this chapter, we consider the expected number of rounds to determine a random n-string when we are allowed to make n queries per round. We present a simple probabilistic analysis that $O(\lg n)$ rounds suffice for random strings with high probability.

The key issue in this kind of analysis is the probability that an arbitrary l-string is a substring of a random n-string. Because of clustering effects for low-period strings, (for example, the string 0^k is likely to occur more than once in a binary string if it occurs at all) the probability that a given string s occurs in a random n-string is a function of s, not just the length of s. Guibas and Odlyzko [5] and Wilf [16] use generating function methods to count the number of n-strings containing a substring s. However, simple counting arguments show that the probability goes to zero for l-strings where $l \geq (1+\epsilon)\log_\alpha n$ and to one for $l \leq (1-\epsilon)\log_\alpha n$.

Theorem 16 *Let* S *be a random* n*-string on an alphabet of size* α. *With a probability of* $1 - 1/n^\epsilon$, S *can be determined using* $O(\alpha \cdot \epsilon \log_\alpha n)$ *rounds of* n *queries per round.*

Proof: We will use a three-phase strategy to determine S. First, we use one round of n queries to implement the classical sequencing chip $C(\lfloor \log_\alpha n \rfloor)$, thus determining all $\leq n$ distinct $(\log_\alpha n)$-substrings of S. Second, we will use $\alpha \cdot \epsilon \log_\alpha n$ rounds to 'grow' each of these strings to length $l = (1+\epsilon)\log_\alpha n$ using the technique of Lemma 8. Finally, we perform the doubling strategy of Lemma 12 to complete the determination of S, starting from the set of l-substrings.

The remaining issue is to analyze the number of questions asked in the first round of the third phase. Since $O(n)$ of the concatenations correspond to actual $2l$-substrings of S, all of these questions must be asked, plus any of the $O(n^2)$ 'false' questions which happen to have all l-substrings occur in S.

Suppose we just refrain from asking the 'false' questions xy whose central l-substring s is not in S. There are three different cases where s is in X but xy is not – (1) the $l/2$ characters after x form s with x, (2) the $l/2$ characters before y form s with y, or (3) s occurs elsewhere in S, not flanked by x or y. Cases (1) and (2) each occur with probability $\alpha^{l/2}$, while case (3) occurs with probability α^l. Thus the expected number of 'false' questions to survive to the first doubling is $2n^2/n^{(1+\epsilon)}$, which is sublinear for $\epsilon > 1$. Thus an expected $O(n)$ questions need to be asked in the first doubling round, which can simulated using a constant number of rounds of n questions. Further, the expected number of false questions decreases in subsequent doubling rounds, so $O(\lg n)$ rounds of n questions suffices for this last stage. ▮

In fact, it is obvious that fewer rounds on average should suffice, since the concatenation of two l-strings should go unasked if *any* of its l-substrings is not in S, instead of just the middle one. The lack of independence makes the analysis of this difficult, however, our simulation results in the appendix shows that the improvement is considerable.

6 Conclusions

We conclude with a list of open problems:

- Give improve our upper bounds or prove interesting lower bounds on capacity for the cases of few rounds. We have presented tight lower bounds for the cases of one round and one question per round, but nothing interesting in between.

- Generalize these results to sequencing multiple target strings, ie. where we are simultaneously sequencing many strings. This approach is applicable to Crkvenjakov and Drmanac's target down approach to sequencing by hybridization [3, 11], which makes one query per round but achieves parallelism through multiple targets.

- Generalize these results for the case of positive and negative errors.

- Design more efficient sequencing chips (in the average case), and reconstruction algorithms for them.

- Does our interactive model have a practical laboratory implementation? The issue is not one of feasibility, but of cost, since either the photolithography methods of [4] or the primer walking technique of [7] can be used to realize it, albeit in an expensive manner.

Acknowledgements

We would like to thank Sundaram Gopalakrishnan, Rob Lipshutz, and Bill Sofer for useful discussions.

References

[1] L. M. Adleman. Molecular computations of solutions to combinatorial problems. *Science*, 266:1021–1024, November 11, 1994.

[2] N. Broude, T. Sano, C. Smith, and C. Cantor. Enhanced DNA sequencing by hybridization. *Proc. National Academy of Sciences*, 1994.

[3] R. Dramanac and R. Crkvenjakov. DNA sequencing by hybridization. Yugoslav Patent Application 570, 1987.

[4] S. Fodor, J. Read, M. Pirrung, L. Stryer, A. Lu, and D. Solas. Light-directed, spatially addressable parallel chemical synthesis. *Science*, 251:767–773, 1991.

[5] L. Guibas and A. Odlyzko. String overlaps, pattern matching, and non-transitive games. *J. Combinatorial Theory (Series A)*, 23:183–208, 1981.

[6] S. Hannenhalli, P. Pevzner, H. Lewis, and S. Skiena. Positional sequencing by hybridization. Technical Report 95-19, DIMACS, Rutgers University, 1995.

[7] J. Kieleczawa, J. Dunn, and F. Studier. DNA sequencing by primer walking with strings of congiuous hexamers. *Science*, 258:1787–1991, 1992.

[8] R. J. Lipton. Speeding up computations via molecular biology. Princeton University, December 9, 1994.

[9] R. C. Lyndon and M. P. Schutzenberger. The equation $a^m = b^n c^p$ in a free group. *Michigan J. Math.*, 9:289–298, 1962.

[10] P. Pevzner, Y. Lysov, K. Khrapko, A. Belyavski, V. Florentiev, and A. Mizabelkov. Improved chips for sequencing by hybridization. *J. Biomolecular Structure and Dynamics*, 9:399–410, 1991.

[11] P. A. Pevzner and R. J. Lipshutz. Towards DNA sequencing chips. In *19th Int. Conf. Mathematical Foundations of Computer Science*, volume 841, pages 143–158, Lecture Notes in Computer Science, 1994.

[12] P.A. Pevzner. l-tuple DNA sequencing: Computer analysis. *J. Biomolecular Structure and Dynamics*, 7:63–73, 1989.

[13] O. Razgulyaev, A. Rubinov, M. Gelfand, and A. Chetverin. Sequencing potential of nested strand hybridization. *J. Computational Biology*, to appear, 1995.

[14] A. R. Rubinov and M. S. Gelfand. Reconstruction of a string from substring precidence data. *J. Computational Biology*, to appear, 1995.

[15] S. Skiena and G. Sundaram. Reconstructing strings from substrings. *J. Computational Biology*, to appear.

[16] H. Wilf. Strings, substrings, and the nearest integer function. *Amer. Math. Monthly*, 94:855–860, 1987.

Appendix: Simulation Results

We implemented two different algorithms for interactive SBH, and experimented with them on real and simulated DNA. For consistency, we performed n substring queries in each simulated round, where n is the length of the string to be sequenced. To calibrate for smaller size chips, observe that $c \cdot k$ rounds suffice using a chip of size n/c if k rounds suffice for a chip of size n. To calibrate for larger size chips, observe that the number of rounds can only decrease with increasing chip size.

The algorithms we implemented are:

- *The Doubling Algorithm* – This is a direct implementation of the worst-case $O(\lg n)$ round algorithm of Theorem 12. Given the set of all l-substrings of the unknown n-string S, we ask all $2l$-length queries formed by concatenating two substrings together, provided all the l-substrings of the prospective query are l-substrings of S. If there are M_i queries in the ith meta-round which survive this test, they are asked in $\lceil M_i/n \rceil$ rounds of n. In our experiment, the first round starts all questions of length one, and no walking steps (as recommended in the probabilistic algorithm of Theorem 16) were performed.

- *The Adaptive Length Algorithm* – This algorithm is an version of the expected $O(\lg n)$ round algorithm of Theorem 16 which has been enhanced in two ways. First, as in the previous algorithm, a query is pruned unless all the l-substrings of the prospective query are l-substrings of S. Second, and more importantly, instead of always extending the query length by one each round in the critical region, we find the longest l' such that at most $c \cdot n$ length l' queries are consistent with the set of l-substrings of S. Each such meta-round is simulated by at most c rounds of n queries, except for the special case where $l' = l + 1$, and α rounds may be required. Note that l' may grow very rapidly. For example, although our first stage asks only length one questions, the second stage queries are typically $\lg_\alpha n$ in length.

  Certain details are necessary for an efficient implementation of the adaptive length algorithm for long strings. We find that $c = 2$ minimizes the number of rounds for both $\alpha = 2$ and $\alpha = 4$. Also, we use a one-sided binary search to search for l' from l, and a linear-space suffix-tree data structure to quickly establish the necessity of a prospective query.

We have evaluated these algorithms on both simulated and real data. In Figures 2 and 3, we show the number of rounds required for both algorithms to determine random binary and quadrary strings of length 2^i, for $2 \leq i \leq 16$. For each size and algorithm, ten random strings were 'sequenced'. It is clear that the number of rounds required for the adaptive-length algorithm is growing *extremely* slowly, perhaps $O(\lg \lg n)$. The number of rounds is essentially a small

Sequence	Length	Rnds	Queries
Human alpha globin	12,847	12	125,546
Human beta globin	18,060	11	167,722
Chicken collagen	21,180	9	153,836
HIV	9,718	11	83,954
Bacteriophage lambda	48,502	11	386,218
Mouse mitochondrion	16,295	10	120,030
Rat MHC gene	25,759	11	235,652
Rabies virus	11,928	11	99,167
Human rhinovirus type 14	7,212	9	52,634
Human Ribosomal DNA	42,999	16	573,014
Simian Virus 40	5,243	11	48,003
Drosophila white locus	14,245	10	113,202

Table 2: Performance of The Adaptive Algorithm on GenBank Sequences.

constant for imaginable values of n, which bodes well for the potential of interactive SBH.

The number of rounds used by the doubling algorithm demonstrates a startling degree of non-monotonicity, ie. longer strings can require substantially fewer rounds to sequence. This cycling depends upon the value of $\Delta = \lfloor \lg_\alpha n \rfloor - \lg_\alpha n$, as can be illustrated by the difference in periods for binary and quadrary alphabets. Since almost all $\alpha^{\lg_\alpha n}$ ($\lg_\alpha n$)-strings are likely to occur as substrings of S, a large fraction of the $O(n^2)$ possible concatenations will survive (for $\Delta \approx 0$) to be asked as queries in the next round. It is this behavior that the 'walking' steps of Theorem 16 was designed to avoid.

In Table 2, we report on the number of rounds required to determine actual DNA sequences, as drawn from GenBank. The number of rounds required for actual DNA sequences seems to be slightly larger than for random data, presumably because of longer repeat sequences in DNA. However, a dozen rounds suffice to sequence all but one of the DNA sequences in our test, still very modest considering the small sizes of the sequencing chips required. The total number of queries given in Table 2 is less than the number of rounds times the maximum number of questions allowed per round because not all rounds are completely filled.

In fact, the total number of questions over the set of rounds is sufficiently small to justify using these algorithms for the design of *customized* sequencing chips, which seek to identify mutations in specific genes for diagnostic purposes. By making a chip which is the union of all of the queries made over all rounds for a specific sequence, we are guaranteed that any mutation will be detected, since if all questions are answered identically to the original sequence, the test sequence must be identical to the original. Further, we would expect to be able to identify the exact mutation in many cases, from the sequence of differences in the answers.

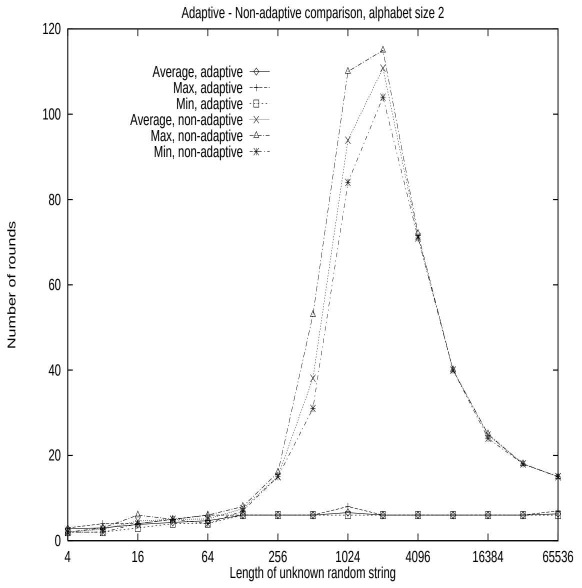

Figure 2: The performance of both algorithms on binary alphabets.

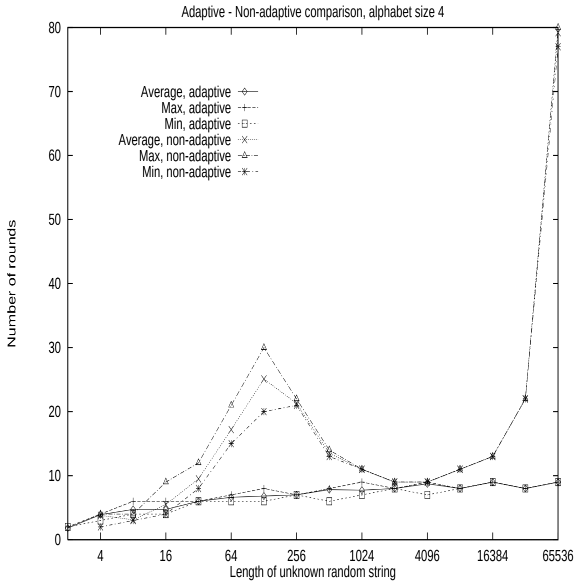

Figure 3: The performance of both algorithms on 4-letter alphabets.