An Optimal Procedure for Gap Closing in Whole Genome Shotgun Sequencing

Richard Beigel[*]
NEC Research Institute & Temple University

Noga Alon[†]
Tel Aviv University

Mehmet Serkan Apaydın[‡]
Stanford University

Lance Fortnow[§]
NEC Research Institute

Simon Kasif[¶]
Cambridge Research Lab., Compaq & The MIT Genome Center

Abstract

Tettelin et. al. proposed a new method for closing the gaps in whole genome shotgun sequencing projects. The method uses a multiplex PCR strategy in order to minimize the time and effort required to sequence the DNA in the missing gaps. This procedure has been used in a number of microbial sequencing projects including *Streptococcus pneumoniae* and other bacteria. In this paper we describe a theoretical framework for this problem and propose an improved method that guarantees to minimize the number of steps involved in the gap closure procedures. In particular, given a collection of $n/2$ DNA fragments we describe a strategy that requires $0.75n \log n$ work in eight parallel rounds of experiments, closely matching a corresponding lower bound of $0.5n \log n$.

1. Introduction

Whole genome sequencing is a revolutionary approach to probing into the genetic make-up of living organisms. The DNA of the first independently living organism, *H. influenzæ*, was sequenced in 1995 by The Institute for Genomic Research (TIGR) [7]. Since, over a hundred genomic projects have been initiated, two dozens completed including a major portion of the human genome. Many complex organisms are also being sequenced, including a variety of mammals, plants, and many microbes and pathogens.

Shotgun sequencing is currently the most widely used approach for whole genome sequencing [11]. It has been used in most microbial projects, Drosophila as well as sequencing the Human Genome at Celera and the mouse genome at the Whitehead Institute. Shotgun sequencing involves the generation of short DNA pieces providing a redundant coverage of the genome. These DNA fragments are subsequently assembled, in-silico, by a computational algorithm. The typical genomic

---

[*] Address: Dept. of Computer and Information Sciences, Wachman Hall Rm 304 (038-24), Temple University, 1805 N Broad St, Philadelphia PA 19122-6094, USA. Research performed in part at DIMACS. Email: beigel@joda.cis.temple.edu. Supported in part by the National Science Foundation under grants CCR-9996021, and CCR-0049019.

[†] Address: Dept. of Mathematics, Tel Aviv University, 69978 Tel Aviv, ISRAEL. Email: noga@math.tau.ac.il.

[‡] Email: apaydin@cs.stanford.edu.

[§] Address: 4 Independence Way, Princeton, NJ 08540. Email: fortnow@research.nj.nec.com.

[¶] Current Address: Dept. of Biomedical Engineering, Boston University, Boston, MA 02215. Email: kasif@bu.edu. Supported in part by NSF.

assembler repeatedly merges DNA fragments with similar (overlapping) ends into increasingly larger fragments (contigs) until no more merging is possible or it is difficult to statistically justify a merge based on the available information. Due to various technical problems related to both biology (e.g., non-clonable sequences or coverage of DNA libraries) and statistics of the sequence (e.g, repeats) some regions of the actual DNA sequence are not covered by the contigs.

This problem creates gaps in the genomic sequence that are often difficult to close. There are several techniques to link across such gaps. A popular approach is using "primer walking", however, this technique is not easily implementable when the length of the gap is large. One powerful method to close a gap relies on a generating a PCR product across the gap. This process requires first producing unique primers at the end of each gap. In particular, we need to choose primers that correspond to regions outside repeats in the currently sequenced contigs (see Figure 1). Then the PCR products are "walked" across the sequence until they "meet" and create a reaction that can be observed in a tube. In order to test whether a particular pair of oriented contigs might be adjacent in the genome, we need to place the two primers corresponding to the two ends and the genomic sequence into a tube where the reaction can be observed.

The obvious approach to combinatorial PCR will test every pair of ends, thus, creating $O(n^2)$ tests (tubes). For a large number of gaps this is not feasible. An alternative approach was proposed and implemented in the lab by [10]. The approach is based on a multiplex PCR [5] where multiple primers are pooled together and then tested simultaneously. To illustrate a simple version of multiplex PCR consider as an example the problem of pairing $N = 100$ primers. The other critical parameter that we need to consider is the maximum number or primers that can be placed in a tube.

In the example, we will assume this number is $K = 20$. We first create 10 pools of primers, with 10 primers per pool. We subsequently pair each pool using $\binom{N}{2}$ tubes. In order to create this experiment we assume we start from a state where each primer is placed in a tube. Then we use pipetting to create the pools. In this case we use 100 pipetting operations to place the 100 primers into the 10 pools first. Then we use additional 90 pipettings to place the mixed pools into the reaction tubes (45 tubes with two pools per tube) we need 90 pipettings. The entire process requires 190 pipetting operations.

This approach is hardly optimal in terms of the number of reaction tubes required, and in fact [10] propose a more sophisticated approach based on block design (affine planes) which guarantees to minimize the number of tubes needed for pairing all the primers. The multiplex PCR method was used for the closing the sequence the genomes of *Streptococcus pneumoniae*, *Shewanella putrefaciens*, *Staphylococcus aureus*, and *Chlorobium tepidum.*

While the multiplex PCR approach has been shown empirically very effective for small number of gaps, it has not been analyzed using a precise theoretical framework that allows to evaluate the optimality or scalability of the technique for a large number of gaps. For example, the number of gaps in the Drosophila genomes is over 400. In many cases a multiplex PCR method creates multiple reactions per tube, in which case we still need to continue the experiments to deconvolve the observed results in order to check which primers exactly created the reactions. In addition, each PCR reaction experiment takes a substantial amount of time, e.g, hours or even an entire day. Thus, another parameter to optimize in addition to number of tubes or number of pipettings is the total time needed to perform the experiment until all possible reactions have been identified. In general, we expect that future PCR experiments will be carried by robotic devices where the overall time and the total work are the main critical resources.

In this paper we develop a natural theoretical framework for multiplex PCR that allows us to formulate the problem in computational terms. In particular, we provide a formalism that allows to minimize the total number of PCR tests (called work) as well as the completion time. We formulate a version of the problem where we perform PCR experiments in parallel rounds; the time for the entire process is proportional to the number of rounds. Throughout log denotes $\log_2$ and ln de-

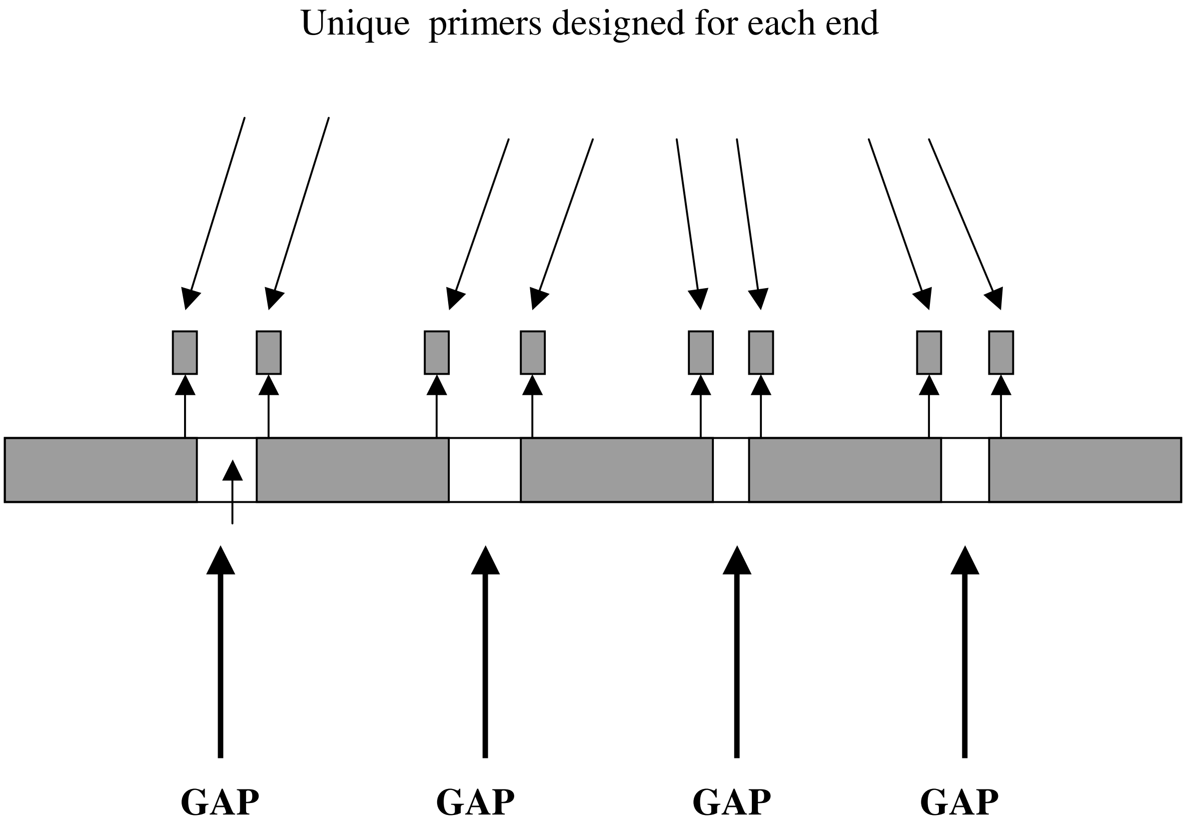

Figure 1: A partially assembled DNA sequence with four gaps requires eight primers to be designed for each end and subsequently "matched" to each other using PCR in order to identify the order and orientation of the contigs in the genome

notes $\log_e$.

2. Learning a Matching

We can think abstractly of our biological problem as follows: Given n chemicals, each of which reacts with at most one of the other n chemicals, determine exactly which pairs of chemicals react. A "test" consists of putting a subset of the chemicals in a tube and seeing if any of them react. Based on the underlying biology, we can assume that there is a reaction in a tube iff at least one pair of chemicals in the tube reacts by themselves.

We call a pair of chemicals that react a *match*. We assume throughout that each chemical reacts with at most one other, i.e., each chemical belongs to at most one match. Given a collection of chemicals, the set of all matches among them is called their *matching*.

3. Parallel Matching-Finding Algorithm

We are given a set S of n chemicals and want to find their matching. The following lemma is key to obtaining a fast parallel algorithm.

Lemma 1 (Partition). *If we partition S into n/s subsets of size s, the expected number of matches internal to subsets is at most $s/2$. In particular, the expected number of subsets that contain a match is at most $s/2$.*

Proof: S contains at most $n/2$ matches. Thus the probability that any particular pair is a match is at most $(n/2)/\binom{n}{2} = 1/(n-1)$. There are $(n/s)\binom{s}{2}$ pairs in the n/s subsets of size s. Thus the expected number of matches internal to them is

$$(n/s)s((s-1)/2)/(n-1) =$$

$$(1/2)n(s-1)/(n-1) \leq s/2$$

since $s \leq n$. ∎

3.1. Finding a Bipartite Matching

In Figure 2, we solve a bipartite version of the matching problem, which is interesting in its own right. Given two sets A and $B = \{b_1, ..., b_{n/2}\}$ of size $n/2$ such that A contains no matches and B contains no matches, find all matches between A and B. We will reduce the general matching problem to the bipartite case.

Since $\ell \leq \log n$, this can be done in 1 round with $0.5n \log n$ work.

3.2. Partitioning into Matchless Subsets

In Figure 3 we reduce to the bipartite case by partitioning S into several subsets, each of which contains no matches. The number c was chosen so as to minimize the number of tests performed by the algorithm.

Step 1: We start by performing $n^{0.7}$ tests. By the lemma, the expected number of pieces that contain a match is $(1/2)n^{0.3}$. The brute force comparison takes time $\binom{n^{0.3}}{2}$ per piece. So the total expected work for step 1 is at most $n^{0.7} + (1/4)n^{0.9}$. The time for Step 1 is 2 rounds.

Step 2: We start by performing $\sqrt{n/c}$ tests. By the Partition Lemma, the expected number of groups that contain a match is at most $(1/2)\sqrt{cn}$. For each group G that contains a match we perform $\binom{\sqrt{c}n^{0.2}}{2}$ tests to see which pairs of pieces in G contain matches. This contributes at most $(1/4)cn^{0.9}$ to the total work.

The expected number of pairs that contain a match is also at most $(1/2)\sqrt{cn}$ by the lemma. Matches between P_1 and P_2 can be found by our bipartite algorithm with work $0.3n^{0.3}\log n$ so this contributes at most $(3/20)\sqrt{c}n^{0.8}\log n$ to the total expected work.

The time for step 2 is 3 rounds

So Partition runs in 5 rounds with expected work at most

$$0.25(c+1)n^{0.9} + 0.15\sqrt{c}n^{0.8}\log n + n^{0.7} + \sqrt{c}n^{0.5}$$

3.3. Putting the Pieces Together

In Figure 4, we present our matching algorithm.

```
For each element $a$ of $A$ do in parallel
    let $\ell = \lceil \log(n/2+1) \rceil$
    /* let $\mathrm{bit}_i(j)$ denote the $i$th bit in the binary representation of $j$ */
    for $i := 0$ to $\ell$ do
        if $a$ reacts with any $b_j$ such that $\mathrm{bit}_i(j) = 1$ then $a_i := 1$ else $a_i := 0$
    let $j$ be the number whose binary representation is $a_\ell ... a_0$
    if $j \neq 0$ then $a$ reacts with the element numbered $a_\ell ... a_0$
```

Figure 2: Procedure BipartiteMatch

```
Let $c = \ln 2$.
Step 1: Partition $S$ randomly into $n^{0.7}$ pieces of size $n^{0.3}$
        for each piece $P$
            if $P$ contains a match then
                by brute force, find all matches in $P$
        remove all matches found
Step 2: Randomly combine pieces $\sqrt{c}n^{0.2}$ at a time to form $\sqrt{n/c}$ groups
        for each group $G$
            if $G$ contains a match then
                for each pair of pieces $P_1, P_2$ in $G$ do
                    if there is a match in $P_1$ union $P_2$ then
                        find all matches between $P_1$ and $P_2$
        remove all matches found
```

Figure 3: Procedure Partition

```
Step 1: Perform procedure Partition $\sqrt{\log n}$ times in parallel to produce $\sqrt{(1/c) n \log n}$ subsets of size
$\sqrt{cn}$ each, none of which contains a match.
Step 2: For each pair $S_1, S_2$ of subsets
            test whether there is a match between $S_1$ and $S_2$
Step 3: For each pair $i, j$ of chemicals
            skip := false
            for every subset $I$ containing $i$
                for every subset $J$ containing $j$
                    if no match was found between $I$ and $J$ then
                        skip := true
            if not skip then
                test whether $i$ reacts with $j$
```

Figure 4: Algorithm Match

Step 1: 5 rounds and $(0.25(c+1)n^{0.9} + 0.15\sqrt{c}n^{0.8}\log n + n^{0.7} + \sqrt{c}n^{0.5})\sqrt{\log n}$ work

Step 2: 1 round and $\binom{\sqrt{(1/c)n\log n}}{2} < 0.5(1/c)n\log n$ work

Step 3: If i reacts with j then clearly we will test it. This case results in $n/2$ tests.

If there is a single subset that contains both i and j then no tests will be performed for the pair i, j.

Otherwise, the probability that every I containing i reacts with every J containing j is at most $\frac{1+o(1)}{n}$. This requires a nontrivial analysis, which we will present in the final version, because we do not have complete independence on the tests. This case results in $\binom{n}{2}/n < n/2$ tests.

Step 3 takes only 1 round.

Thus Algorithm match runs in 7 rounds and does $0.5(1/c)n\log n + n + o(n) = (1/(2\ln 2))n\log n + n + o(n)$ work. This is asymptotically less than $0.72135n\log n$.

4. Lower Bound and a Serial Algorithm

We prove that the matching problem requires $0.5n\log n - O(n)$ tests, even if they are performed sequentially. Thus our parallel algorithm is within a factor of ln 2 of optimal. We also present a serial algorithm that uses only $0.5n\log n + n$ tests, and is therefore optimal up to first order.

4.1. Lower Bound

We will actually show the lower bound holds for even for the simpler bipartite matching case described in Section 3.1. We use a proof similar to the $\Omega(n\log n)$ lower bound for sorting.

Each experiment we perform produces only two possible outcomes: we find a reaction or we don't. Let $S_{i_1 i_2 \ldots i_k}$ for $i_j \in \{0,1\}$ be the set of matchings on a bipartite graph that on the jth experiment produces a reaction iff $i_j = 1$.

Suppose we only need k experiments. This means that for each $\vec{i} \in \{0,1\}^k$, $S_{\vec{i}}$ has at most one matching. The total number of possible matchings is 2^k or, equivalently $k \geq \log m$ where m is the number of matchings on a bipartite graph.

Each matching on a bipartite graph corresponds to a permutation of the $n/2$ vertices in one side of the graph. So we have $m = \frac{n}{2}!$. By Stirling's approximation to the factorial we have $k \geq \log m \approx \frac{n}{2}\log n - \theta(n)$.

4.2. Optimal Serial Algorithm

Given a set $\{c_1, \ldots, c_n\}$ of chemicals, we will determine their matching.

```
Algorithm SerialMatch
$T := \{\}$
for $i := 1$ to $n$ do
    $T := T \cup \{c_i\}$
    if there is a reaction in $T$ then
        $T := T - \{c_i\}$
        find which element $c_j$ in $T$ reacts with $c_i$
        $T := T - \{c_j\}$
End of Algorithm SerialMatch
```

The "find" step can be performed with $\log n$ parallel tests, using the binary representation trick from our bipartite matching algorithm. The algorithm takes $(3/2)n$ rounds and makes a total of $n + (n/2)\log n$ tests.

5. Simulation

We implemented our proposal in a simulation. We wanted to see whether our theoretical analysis is supported in an experimental setting as well as providing lab biologists with an effective tool to guide the experiments. The software is available from the corresponding author by email request (kasif@genome.wi.mit.edu). Our simulation supports the theoretical analysis. The observed work is very close to the predicted estimates with relatively little variance. The results are given in Figure 5.

6. Discussion

Combinatorial algorithm design has been playing a consistently important role in sequencing, mapping, assembly, DNA chip design and analysis,

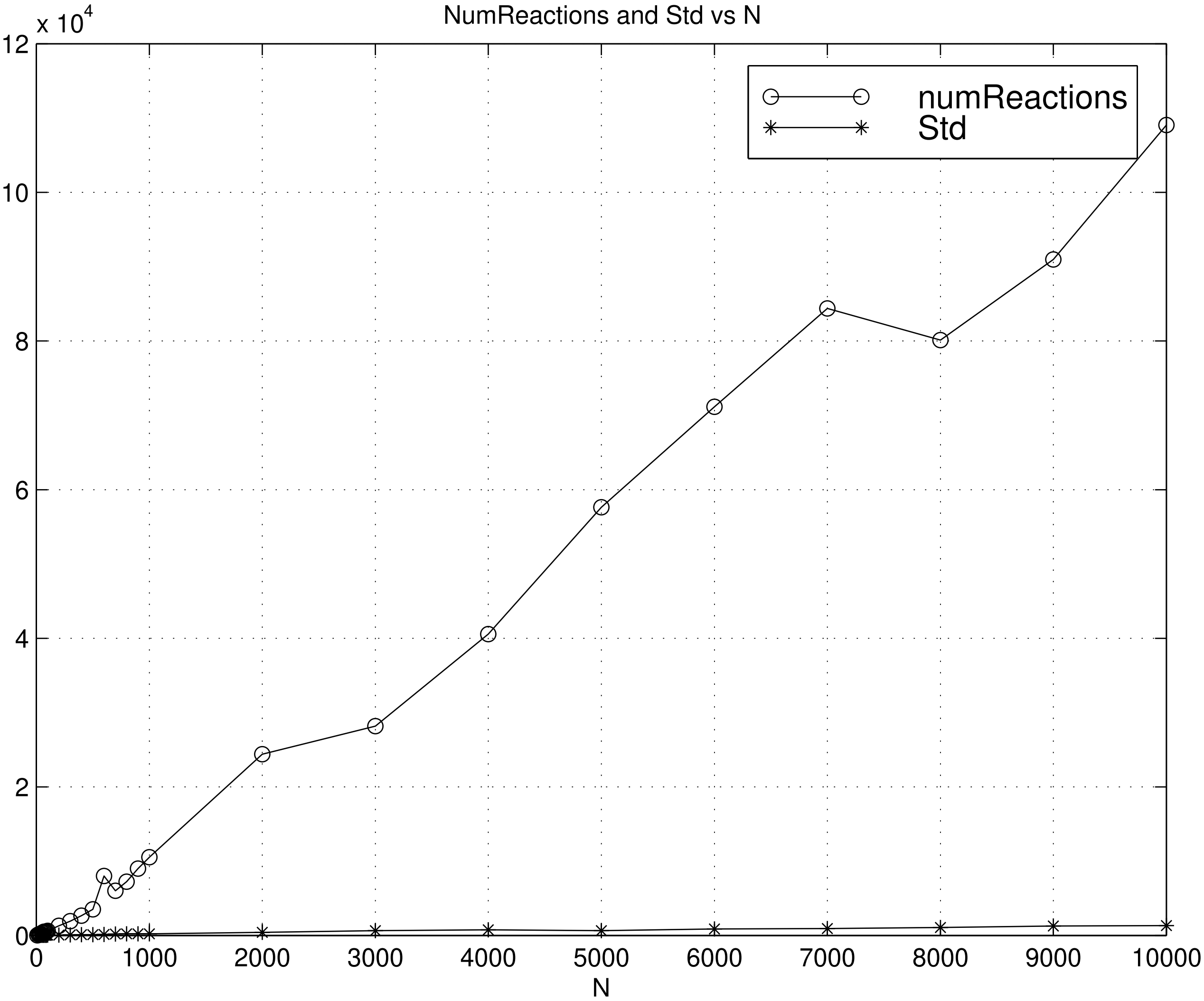

Figure 5: A Graph of the number of reactions for a given number of ends N and the corresponding variance

linkage analysis, protein clustering and many other problems [6, 9, 12, 8, 1]. In this paper we provided a theoretical framework for analyzing the problem of closing gaps in whole genomes. We also devised an effective method with very good theoretical and empirical performance (in a simulation). The general problem addressed in the paper, namely efficiently pairing of objects is related to a number of other important problems in computational biology. For example, screening experiments against a DNA library requires efficient pooling strategies [3, 2, 4]. DNA pooling of a rather different nature has been also used in genetic tracking of genes for complex traits. There are several open theoretical questions.

- Can we solve the matching problem in time $\alpha n \log n + o(n \log n)$ for some $\alpha < 1/(2 \ln 2)$?
- Can we make Step 2 of Algorithm Match deterministic (perhaps by using block designs)?
- Can we make the algorithms described in the paper robust to errors that sometimes sneak into experimental laboratory procedures? A relatively simple solution to cope with false negatives, namely missed reactions, is to retest the primers that were not matched by a recursive application of the algorithm we described. This approach is naturally not optimal when the error is high. False positive errors, namely spurious reactions, can be eliminated by retesting of the matched pairs. Again, the efficiency of the solution will be highly dependent on the relative error rate. We expect the false positive rate to be small assuming the primers are well designed. Once we have a more accurate estimate on the number of errors in large-scale gap-closing procedures we can adapt our algorithm to address this problem.
- The problem we studied is theoretically equivalent to the problem of learning read-once monotone 2DNF formulas with only membership queries. Is there a computational learning formalism that captures this problem especially with respect to the anticipated noise in multiplex PCR procedures?

We also plan to implement our proposal in the lab in one of the ongoing genome sequencing projects. The previous version of multiplex PCR is now used routinely at TIGR. However, the previous method relies on a heuristic procedure with substantial human assistance in inspecting the reaction tubes. For a large number of gaps, a robotic implementation is essential. We believe the current proposal is likely to result in an improved overall procedure, especially if assisted by a fully automated robotic installation similar to the ones used at the Whitehead Institute and other major genome centers.

References

[1] F. Alizadeh, R. Karp, D. Weisser, and G. Zweig. Physical mapping of chromosomes using unique probes. In *Proc. 5th ACM-SIAM Symposium on Discrete Algorithms*, pages 489–500, 1994.

[2] D. J. Balding, W. J. Bruno, E. Knill, and D. C. Torney. A comparative study of non-adaptive pooling designs. In T. Speed and M. S. Waterman, editors, *Genetic Mapping and DNA Sequencing*, volume 31 of *IMA Volumes in Mathematics and its Applications*, pages 133–155. Springer-Verlag, Berlin, 1998.

[3] W. J. Bruno, E. Knill, D. J. Balding, D. C. Bruce, N. A. Doggett, W. W. Sawhill, R. L. Stallings, C. C. Whittaker, and D. C. Torney. Efficient pooling designs for library screening. *Genomics*, 26:21–30, 1995.

[4] W. J. Bruno, F. Sun, and D. C. Torney. Optimizing non-adaptive group tests for objects with heterogeneous priors. *SIAM J. Appl. Math.*, 58:1043–1059, 1998.

[5] L.J. Burgart, R.A. Robinson, M.J. Heller, W.W. Wilke, O.K. Iakoubova, and J.C. Cheville. Multiplex polymerase chain reaction. *Mod. Pathol.*, 5:320–323, 1992.

[6] C. Cantor, P. Gillevet, R. Karp, G. Myers, M. Olson, P. Pevzner, and M. Yannakakis, editors. *Combinatorial Methods for DNA Map-*

ping and Sequencing. Rutgers University, Rutgers, New Jersey, October 1994.

[7] R.D. Fleischmann, M. Adams, O. White, R. Clayton, E. Kirkness, A. Kerlavage, C. Bult, J.-F. Tomb, B. Dougherty, J. Merrick, K. McKenney, G. Sutton, W. FitzHugh, C. Fields, J. Gocayne, J. Scott, R. Shirley, L.-I. Liu, A. Glodek, J. Kelley, J. Weidman, C. Phillips, T. Spriggs, E. Hedblom, M. Cotton, T. Utterback, M. Hanna, D. Nguyen, D. Saudek, R. Brandon, L. Fine, J. Fritchman, J. Fuhrmann, N. Geoghagen, C. Gnehm, L. McDonald, K. Small, C. Fraser, H. Smith, and J.C. Venter. Whole-genome random sequencing and assembly of *Haemophilus influenzae Rd. Science*, 269:496–512, 1995.

[8] D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge University Press, New York, 1997.

[9] P. Pevzner and M. Waterman. Open combinatorial problems in computational molecular biology. In *Proc. Israel Symposium on Systems and.* IEEE Computer Society Press, January 1995.

[10] H. Tettelin, D. Radune, S. Kasif, H. Khouri, and S.L. Salzberg. Optimized multiplex pcr: Efficiently closing a whole-genome shotgun sequencing project. *Genomics*, 62:500–507, 1999.

[11] J.C. Venter, M.D. Adams, G.G. Sutton, A.R. Kerlavage, H.O. Smith, and M. Hunkapiller. Shotgun sequencing of the human genome. *Science*, 280(5369):1540–1542, 1998.

[12] M. Waterman. *Introduction to Computational Biology: Maps, Sequences, and Genomes.* Chapman & Hall, New York, 1995.