The Restriction Scaffold Problem

Amir Ben-Dor [*] Richard M. Karp [†] Benno Schwikowski [‡] Ron Shamir [§]

ABSTRACT

Most shotgun sequencing projects undergo a long and costly phase of finishing, in which a partial assembly forms several contigs whose order, orientation and relative distance is unknown. We propose here a new technique that supplements the shotgun assembly data by cheap and simple complete restriction digests of the target. By computationally combining information from the contig sequences and the fragment sizes measured for several different enzymes, we seek to form a "scaffold" on which the contigs will be placed in their correct orientation, order and distance. We give a heuristic search algorithm for solving the problem and report on promising preliminary simulation results. The key to the success of the search scheme is the very rapid solution of two time-critical subproblems that are solved precisely in linear time.

Our simulations indicate that with noise levels of some 3% relative error in measuring fragment sizes, using five enzymes, most datasets of 20 contigs can be correctly ordered, and the remaining ones have most of their pairs of neighboring contigs correct. Hence, the technique has a potential to provide real help to finishing. Even when the target clone remains unfinished, the ability to order and orient the contigs correctly makes the partial assembly both more accessible and more useful for biologists.

[*]Life Science and Technology Laboratory, Agilent Technologies. E-mail: amir_ben-dor@agilent.com.
[†]Department of EECS, University of California, Berkeley and International Computer Science Institute, Berkeley, California. E-mail: karp@ICSI.Berkeley.edu.
[‡]The Institute for Systems Biology, 1441 North 34th Street, Seattle, WA 98103-8904, USA. E-mail: benno@systemsbiology.org.
[§]School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel. E-mail: rshamir@tau.ac.il.

1. INTRODUCTION

In this paper we propose a novel approach that combines computation and experimental data to help in the challenge of "finishing" sequencing projects. One of the common strategies for sequencing large clones (e.g., cosmids or BACs) is shotgun sequencing. Larger genomes are often sequenced clone-by-clone (cf. [1].) In this strategy, the researchers clone short subfragments of the target, and then apply to each clone a sequencing reaction, which produces its sequence, also called a *read*. In the shotgun process, read locations along the target are random and initially unknown. Typical read lengths are 500-700 bases. Subsequently, the reads are put together computationally using the overlaps between them, in a process called *sequence assembly*. Moreover, reads may come from both strands of the DNA, and their "strandedness" (orientation) is unknown. Since successful sequence assembly requires substantial read overlaps, high redundancy of reads is needed. In order to obtain almost complete coverage of the target and to be able to assemble the sequence, a high redundancy of 5-10 is needed.

Sequence assembly "glues" together reads that have large overlaps and forms contiguous sequence stretches, called *contigs*. Before the sequence has been completely determined, the solution is a collection of contigs whose relative position, orientation and distances are unknown. It is this information that we set out to compute.

For many applications, achieving high redundancy is too expensive and even unnecessary. For example, in a gene hunting project, one typically would like to obtain the target gene in a BAC, and there is no need to fully assemble all of the reads. If the gene falls into few contigs then studying these contigs may be enough. In another situation, only a rough draft of the target, of relatively low redundancy, may be affordable.

Consider the situation when the assembly process has produced several contigs covering together most (say 95% or more) of the target. The sequence content of the contigs is quite accurate, but the relative positions of the contigs and their orientation (straight or reverse-complemented) are undetermined by the assembly. The "finishing" phase of a sequencing project aims to orient and order the contigs and complete the missing sequences between neighboring contigs (cf. Figure 1.) Determining the correct ordering and orientation of the contigs, and the distances between successive contigs, would greatly expedite the finishing.

Unlike the shotgun phase that is highly automated, finishing is a slower and more laborious process [2]. For example, in the public human genome project [5], where each BAC clone is shotgun-sequenced separately, thousands of BACS are yet to be finished. Due to the importance of sequencing for current biological and medical research, and the large resources put into sequencing, many

approaches have been suggested for expediting finishing. These include:
• Software systems to help with the choice of finishing reads and automate decisions [2];
• "The double barrel shotgun" strategy, which uses information on pairing of reads at approximately known distance [6, 10] (this is the strategy at the basis of the whole genome shotgun approach used for the commercial human genome project [11, 9]);
• Obtaining targeted reads designed to close the gaps between contigs. This is often a very expensive process.
• Performing PCR experiments with primers taken from the ends of two contigs. Since the order and orientation of the contigs is unknown, this requires a lot of trial and error to match the right contig ends. Recently, PCR multiplexing was suggested to ease this problem [8].

In this paper we explore an alternative, cheaper approach in which we completely digest the target with several restriction enzymes (typically 4–6 6-cutters), and obtain for each enzyme the list of restriction fragment sizes contained in the target. Then it may be possible to infer the position and orientation of each contig from the combination of the known sequences of the contigs and the restriction fragment data. The restriction data is thus used as the "scaffold" for placing and orienting the contigs. For this reason, the reconstruction problem is called the Restriction Scaffold Problem (RSP).

We shall give a heuristic search algorithm for solving the RSP and report on some promising preliminary simulation results. The key to the success of the search scheme is very rapid solutions of two subproblems that are solved precisely in linear time. Those subproblems are solved at the innermost loop of the search and are thus repeated millions of times, so efficiency in solving them is paramount. Our simulations indicate that with noise levels of some 3% relative error in measuring fragment sizes, using five enzymes, most contig adjacencies in datasets of 13 contigs were correctly determined. Hence, the technique has a potential to provide real help to finishing using cheap gel experiments and computations. Even when the target clone remains temporarily unfinished, the ability to order and orient the contigs correctly makes the partial assembly both more accessible and more useful to biologists.

The paper is organized as follow: Section 2 contains preliminaries and definitions. Section 3 outlines the algorithm, followed by detailed description of algorithms used for efficient solution of the two subtasks in Sections 4 and 5. We describe simulation results in Section 6. For lack of space, some proofs are sketched or deferred to appendices.

2. PROBLEM FORMULATION

The input to our problem consists of two types of data:

- For each enzyme i, a set $m(i)$ of positive real numbers, representing the measured sizes of the restriction fragments obtained in a complete digest of the target by enzyme i.

- For each contig j and enzyme i, a vector $c_j(i)$, giving the sequence of computed restriction fragment sizes resulting from the complete digestion of contig j by enzyme i, in the order of their distance from a reference end of contig j. This vector is computed from the sequence of contig j, together with knowledge of the recognition sequence for enzyme i. (We assume that the recognition sequences are palindromic, i.e., identical to their reverse complements, as is the case in most enzymes.)

Note that for this abstraction we assume that the sequences of the contigs are known precisely as otherwise computing the vectors of fragment sizes may be incorrect. We now introduce the notion of a solution to RSP. Assume there are n contigs. Any solution must orient and order the contigs, and also determine the sizes of the gaps between successive contigs. A *configuration* (short for configuration of contigs) is a pair $S = (\pi, g)$ where π is a signed permutation of $\{1, \ldots, n\}$ and $g = (g_1, \ldots, g_{n+1})$ is a vector of real numbers. (a *signed permutation* is a permutation in which each number has a $+$ or $-$ sign.) The interpretation is that π gives the order of the contigs along the target along with the orientation for each contig (positive for straight, negative for reverse), and g gives the sizes of the gaps, which are the segments of the target not covered by contigs, in the order of their occurrence along the target. Thus g_i is the size of the gap just preceding the ith contig in the ordering, and g_{n+1} is the size of the gap following the last contig in the ordering. Note that gap sizes can be negative if short overlaps of contigs were not detected. We concentrate on the case where there are no restriction sites in the gaps. Since we are focusing on clone sequencing projects in the finishing phase, where the vast majority of the target is covered by contigs, this is a reasonable assumption. However, we also sketch how our algorithm can be extended to handle the possibility of restriction sites in the gaps.

A contiguration (see Figure 1) positions the contigs along the target. It also determines, for each enzyme, the sizes of the fragments between consecutive cut sites along the target. These include fragments that have endpoints in different contigs. We call such subfragments *bridges*. A bridge usually contains end parts of neighboring contigs, but it may sometimes contain full contigs that happen to have no cut sites for the enzyme. In the absence of restriction sites within bridges, one can compute from a contiguration S the multiset of all restriction fragment sizes for enzyme i, including both bridges and fragments internal to a single contig. We denote that multiset by $F_S(i)$.

Suppose that, for a given contiguration we have determined, for each enzyme i, the following information: $m = m(i)$, the set of measured fragments for enzyme i, and $f = F_S(i)$, the multiset of computed fragments. Note that m is a set rather than a multiset because in the gel electrophoresis experiments that determine the fragment sizes, one cannot accurately distinguish the exact number of fragments that contribute to the same band. Consequently, we do not know the multiplicities of measured fragment sizes. The multiset f is computed from the positions and orientations of the contigs, together with the ordered sequence of computed restriction fragment sizes within each contig.

In the ideal case where the contiguration is correct and all measurements and computations are exactly correct, the set of distinct values in f should be exactly m. In practice, this ideal situation will never occur because of imprecise gel measurement and computation errors, but we expect that, if the contiguration is chosen correctly, then the set m and the multiset f should resemble each other closely. In particular, we would expect that each element of f is close to some element of m, and each element of m is close to some element of f.

We assume a prespecified cost function $Q(f, m)$ for each pair (f, m), that measures the degree of disagreement between f and m. We now describe how the overall cost is assessed. An *assignment* is a

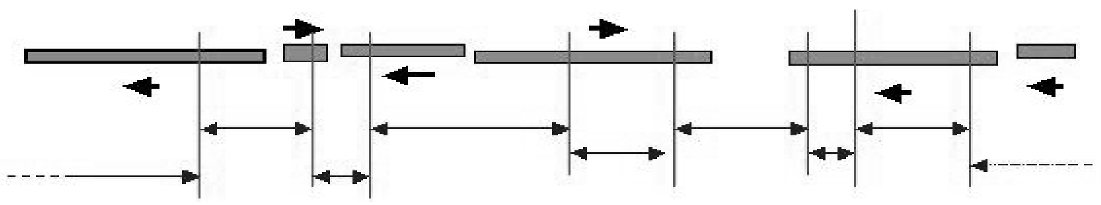

Figure 1: Fraction of a contiguration and one enzyme data. Top: the contigs; bold arrows denote the orientation of each contig. Vertical lines: restrictions sites. Bottom: restriction fragments between neighboring sites.

function from f onto m. Thus an assignment associates a measured fragment with each computed fragment, while requiring that each measured fragment is associated with at least one computed fragment. The *cost* of assignment A is $\sum_{x \in f} Q(x, A(x))$. The cost associated with a particular enzyme that has data f, m is the least value of an assignment between them. The cost of a contiguration is the sum of the costs associated with the different enzymes. We are finally ready to define the main formal problem of this paper:

Restriction Scaffold Problem (RSP):
Given measured and computed fragments, find a contiguration of minimum cost.

3. OVERVIEW OF THE ALGORITHM

In this section we present an overview of the algorithm we employ for the RSP. A complete description is given in the following sections.

In searching for the best contiguration we must determine three components:

1. A signed permutation π;
2. A gap vector g associated with π;
3. For each enzyme, an assignment from the multiset f of computed fragment sizes onto the set m of measured fragment sizes.

The problem is NP-hard even in very degenerate variants (see Appendix A). Two challenges present themselves. First, the space of signed permutations is too large ($n! \cdot 2^n$) for enumeration even for a very moderate n. Second, it is not clear how to find the best gap vector for a given π, since this involves two optimization problems that depend on each other: choosing the gap vector g, and choosing the assignment A.

The approach we take in this paper is a combination of a local search heuristic (namely, Simulated Annealing [4]) that searches in the space of signed permutations, and two very efficient algorithms, one that optimizes gaps (given π and an assignment A), and one that optimizes the assignment (given π and the gaps g). We employ these two algorithms iteratively until convergence. An outline of the algorithm is described in Figure 2. A *reversal* operation on a signed permutation reverses the order and signs of numbers in a contiguous segment.

The algorithm can be extended to deal with restriction sites within bridges. Such a restriction site for enzyme i can be modeled as a fictitious contig of length zero containing a restriction site for enzyme i and no other restriction sites. Since the number of restriction

```
Start from a random signed permutation $\pi$
Let $t$ denote a starting temperature
Repeat until convergence:
    Let $\pi'$ be obtained from $\pi$ by a reversal.
    Start with a uniform gap vector $g$
    Repeat until convergence:
        Find an optimal assignment $A$, given $\pi$ and $g$
        Find an optimal gap $g$, given $\pi$ and $A$
    Let $\delta$ denote $cost(\pi') - cost(\pi)$.
    If $\delta \leq 0$ then $\pi \leftarrow \pi'$.
    Otherwise, $\pi \leftarrow \pi'$ with probability $exp(-\delta/t)$
    Update $t$ according to the cooling scheme.
Report $\pi$
```

Figure 2: Outline of the algorithm

sites within bridges is initially unknown, the algorithm must provide for the operations of inserting or deleting a fictitious contig, as well as the operation of reversal, at the step where contiguration π' is obtained from contiguration π. Since the number of restriction sites within bridges is expected to be small, the cost of a contiguration should include a penalty term which increases sharply with the number of fictitious contigs. This extension of the algorithm has not yet been implemented.

In order to make the algorithm even more robust we repeat the local search described above a number of times, each time starting from a different random signed permutation. In addition to reporting the best contiguration discovered, we rank the contig adjacencies (between pairs of signed contigs) according to how many times they occur in the returned solutions.

Our detailed description of the algorithm is organized as follows. In Section 4 we describe an efficient assignment algorithm (given π and g). Finally, in Section 5 we describe how to efficiently compute gaps (given π and an assignment A).

4. COMPUTING AN OPTIMAL ASSIGNMENT

In this section we present an efficient algorithm, which, given a contiguration (π, g), computes, for each enzyme, an optimal assignment mapping the set of computed fragments f onto the set of measured fragments m. In the next section we describe how to optimally choose the gap vector g, given the signed permutation π and, for each enzyme, a fixed assignment of bridges to measured fragments. Each of these optimization problems requires a cost function Q. We assume that the function Q should satisfy the following properties:

- $Q(x, x) = 0$;
- if the interval $[x, y]$ contains the interval $[x', y']$ then $Q(x, y) \geq Q[x', y']$;

- (order-preserving property) if $x_1 < x_2$ and $y_1 < y_2$ then $Q(x_1, y_1) + Q(x_2, y_2) \leq Q(x_1, y_2) + Q(x_2, y_1)$.

It would be natural to use the same cost function in both optimization problems. This would ensure convergence to a limit in which the gap vector is optimal for the assignment and the assignment is optimal for the gap vector. Instead we use slightly different cost functions in the two optimization problems. This slight inconsistency enables us to achieve a vast increase in computational speed, at the cost of sacrificing the theoretical convergence property.

We shall give a linear-time algorithm for computing an optimal assignment when the function Q is chosen in a particular way. We begin by presenting and solving an abstract assignment problem. We then show how computing the score of a contiguration can be reduced to a special case of this problem, for which a linear-time algorithm is available.

4.1 An Assignment Problem

We consider the following *assignment problem*: given a set of *girls* $G = \{g_1, g_2, \cdots g_t\}$ and a set of *boys* $B = \{b_1, b_2, \cdots, b_s\}$ with $t \geq s$, and a cost function Q from $G \times B$ to the reals, find a function F from G onto B to minimize $\sum_{i=1}^{r} Q(g_i, F(g_i))$. Note that the function F assigns every girl to exactly one boy, and every boy to at least one girl.

By formulating this problem as a minimum-cost flow problem and using known properties of that problem we can show that the problem can be solved by the following two-stage process.

Stage 1: Matching

Construct a matching of minimum cost in which each girl is matched with at most one boy and each boy is matched with exactly one girl, where the cost of matching g_i with b_j is $Q(g_i, b_j)$.

Stage 2: Assigning the Unmatched Girls

For each girl g_i, define $F(g_i)$ as follows: if g_i is matched with b_j then $F(g_i) = b_j$; otherwise $F(g_i) = b_k$ where $C(g_i, b_k) = \min_j C(g_i, b_j)$. Thus the assignment assigns each unmatched girl to her closest boy.

4.2 Linear-Cost Matching

The *linear-cost matching problem* is a special case of the matching problem to be solved in Stage 1. In this special case the boys and girls are mapped to points on the real line, the number of girls is greater than or equal to the number of boys, each girl is to be matched with at most one boy, each boy is to be matched with exactly one girl, and the cost of matching girl i with boy j is the distance between the corresponding points. The cost function is defined more formally as follows. Let S be the union of the set of girls G and the set of boys B. Note that S contains at least as many girls as boys. Associated with each element $e \in S$ is a real number $p(e)$ called the *position* of e. For each girl g_i and boy b_j, $C(g_i, b_j) = |p(g_i) - p(b_j)|$.

A 1975 paper by Karp and Li [3] gives an efficient algorithm for solving the linear-cost matching problem. It is based on the following observations. Suppose the elements of S are sorted in increasing order of their positions (thus, if $p(e_1) < p(e_2)$, then e_1 precedes e_2 in the ordering). For each element e, let $L(e)$ be the number of boys preceding e in the ordering minus the number of girls preceding e in the ordering. Define the *level* of girl g_i as $L(g_i)$ and the level of boy b_j as $L(b_j) + 1$. The following facts can be shown:

- Within each level, the number of girls is either equal to the number of boys or one greater than the number of boys, and the boys and girls in each level alternate in the ordering of S.
- There is an optimal matching in which each boy is matched with a girl within the same level.

These facts imply that the following algorithm produces an optimal matching:

(1) Sort S in increasing order of position and determine the level of each element. (2) Within each level, determine a matching of minimum cost in which each girl is matched with at most one boy and each boy is matched with exactly one girl (such a matching can be found in linear time). (3) Construct the union of the matchings at the different levels.

If S has n elements then Step (1) can be carried out in time $O(n \log n)$, and steps (2) and (3) can be carried out in time $O(n)$. In practice, the algorithm is extremely fast. An example of the algorithm is given in Appendix C.

In the application of linear-cost matching to computing the score of a contiguration, it is desirable to require that the matching satisfy the following *order-preserving property*: if $p(g_1) < p(g_2)$, g_1 is matched with b_1 and g_2 is matched with b_2, then $p(b_1) \leq p(b_2)$. There is always a minimum-cost matching satisfying this property, and such a matching can be constructed as follows:

1. Construct an optimal matching;
2. Delete the girls not used in the matching;
3. Construct an order-preserving one-to-one matching between the remaining girls and the boys.

We choose to require the order-preserving property because, among the optimal solutions to the linear-cost matching problem, it singles out the one that minimizes, for every k, the sum of the k largest distances between matched pairs of boys and girls. Thus the matchings it produces are the most plausible ones among the optimal solutions to the linear-cost matching problem.

The overall algorithm for solving the linear-cost assignment problem is as follows:

1. Sort S;
2. Construct an optimal matching;
3. Construct an optimal order-preserving matching;
4. Assign each unmatched girl to her closest boy.

4.3 Computing an Optimal Assignment

As a first attempt at the problem we can take the girls to be the computed fragments and computed bridges, and the boys to be the fragment measurements. Each girl must be matched with one boy, and each boy must be matched with at least one girl (reflecting the fact that each fragment measurement corresponds to at least one physical fragment, but may correspond to two or more physical fragments comprising a single band on the gel). For each girl or boy e we define $s(e)$, the *size* of e, as the size in base pairs of the corresponding computed fragment or fragment measurement.

However, we must also take into account that physical fragments below some detection limit (which we take to be 300bp) may fail to be detected on the gel. Considering this, we allow each girl of size below the detection limit to be assigned to an "imaginary" extra boy of exactly the same size, and each girl above the detection limit to be assigned to an optional boy of size 300bp. In setting up the assignment problem, we also require the constraint that each boy (even the imaginary ones) must be matched with at least one girl. Assuming that the cost of matching a computed fragment with a measurement of exactly the same size is zero, the following trick allows us to enforce this constraint:

1. For each girl of size s below the detection limit, create an additional imaginary girl of size s and an imaginary boy of size s; also, create one imaginary boy and one imaginary girl of size 300bp.
2. Solve the assignment problem on the set of actual boys and girls augmented by the imaginary boys and girls defined above.
3. Remove the imaginary girls from the assignment, and remove each imaginary boy that is not matched to an actual girl.

In Appendix B we show that this process gives an optimal assignment satisfying the constraint.

It remains to specify the cost function. We define the cost of assigning a girl (computed fragment) g of size $s(g)$ to a boy (measurement) b of size $s(b)$ to be $\ln(\frac{\max(s(g),s(b))}{\min(s(g),s(b))})$. This cost function is an increasing function of the relative error $\frac{\max(s(g),s(b))}{\min(s(g),s(b))} - 1$, is always less than or equal to the relative error, and is a close approximation to the relative error when the relative error is small. The advantage of using this cost function is that the assignment problem can be solved by linear-cost matching, by taking the position $p(e)$ of boy or girl e to be $\ln(s(e))$. With this definition the cost of matching girl g with boy b is exactly $|p(g)-p(e)|$. Thus we can use the extremely efficient linear-cost matching algorithm in computing an optimal assignment.

5. COMPUTING AN OPTIMAL GAP VECTOR

Assume we are given a signed permutation π (specifying the order and orientation of the contigs along the target), and a fixed assignment A. Since A is fixed, the contribution to the cost by each computed fragment that has both ends in the same contig is constant. We focus here on the contribution of the bridges, i.e., computed fragments that span a gap. A assigns each bridge b to an observed length $A(b)$. In this section we show how to compute the gap vector g such as to minimize the deviation between the computed lengths and the measured lengths of the bridges.

If only one restriction enzyme is used, we can trivially choose a gap vector such that the computed length of all bridges would exactly match the corresponding measured length. When more than one enzyme is used, each gap is contained in multiple bridges, and thus a more complex optimization problem arises.

Let us call a bridge *regular* if it spans exactly one gap, and *long* if it spans at least two gaps (and the entire contig between them). Note that a bridge is long only if the corresponding restriction enzyme does not cut certain contigs even once. To allow us to optimize each gap length independently of the others, and thus achieve a linear running time, we choose to optimize the gap lengths only with respect to regular bridges. As most of the target sequence is covered by contigs (typically more than 95%) we expect that only a small fraction of the bridges will be long. Thus, optimal gap lengths with respect to only regular bridges will tend to be near-optimal with respect to all of the bridges.

Consider now a specific gap t that lies between two π-adjacent contigs u and v, and let x denote the length of t. Let $b_1, \ldots, b_k$ be the regular bridges that span t and w.l.o.g. assume that b_i corresponds to enzyme i. Each such bridge contains a subfragment from u, the gap t, and a subfragment from v. Let a_i denote the the sum of the two subfragments for enzyme i. Hence, $b_i = a_i + x$, for each $1 \leq i \leq k$. Let m_i denote the measured length to which b_i is assigned, i.e., $m_i = A(b_i)$.

We define the cost of a bridge b_i, denoted by $c(b_i)$, to be the square of the relative deviation of the computed length of b_i from its measured length:

$$c(b_i) = \left(\frac{|b_i - A(b_i)|}{A(b_i)}\right)^2 = \left(\frac{x + a_i}{m_i} - 1\right)^2$$

The rationale for this definition is twofold. First, we want to penalize big deviations more than linearly, since large relative deviations occur much less frequently in experiments. Second, squaring the deviation eliminates the absolute value function and thus simplifies the optimization. The reason for dividing by m_i is that the error range tends to scale linearly with m_i, rather than being constant.

We define the cost of the gap length x as the sum of the costs of the bridges that span it. That is,

$$c(x) = \sum_{i=1}^{k} c(b_i) = \sum_{i=1}^{k} \left(\frac{x + a_i}{m_i} - 1\right)^2 .$$

By taking the derivative of this objective with respect to x and equating it to zero we obtain:

$$x = \frac{\sum \frac{m_i - a_i}{m_i^2}}{\sum \frac{1}{m_i^2}}$$

6. EXPERIMENTAL RESULTS

To validate our method, we implemented the above algorithm in the programming language C and evaluated its performance on simulated problem instances, for which the discrepancy between computed and correct solutions could be assessed.

Briefly, we generated problem instances by simulating a shotgun sequencing process in which reads are positioned independently and uniformly along the target sequence until a predefined redundancy is achieved. Whenever two reads or contigs overlap sufficiently, the overlap causes the corresponding sequences to be merged. For simplicity, we assumed a conservative read length of 500, but also that contigs were merged whenever their intervals on the target sequence overlapped by any amount. This process results in a number of contigs whose contiguration is known.

As target sequence we used a subsequence of 307862 bases from the human AML1-CBR region on chromosome 21 (GenBank accession number AJ229043). In a simulated restriction digest, we used the six restriction enzymes `Sca I`, `BamH I`, `Sma I`, `Afl II`, `EcoR V`, and `ApaL I`. From the fragment sizes that a complete digest experiment would produce for each enzyme, we obtained measured fragment sizes by adding a normally distributed relative error with mean equal to the precise fragment length and a standard deviation of 3%.

We generated problem instances for three different values of redundancy: 5x, 5.5x, and 6.5x. Using our simulator, we found that average numbers of contigs for these redundancies are approximately 20, 13, and 6, respectively. For simplicity, we restricted our experiments for each redundancy value to instances that contained exactly its corresponding average number of contigs.

Recall that our algorithm returns a ranked list of adjacencies for each problem instance. The quality of such a ranking can be evaluated by an *ROC curve* [7]. An ROC curve represents the different tradeoffs between false and true positive rates that are achieved by using the first $k = 1, 2, \ldots$ adjacencies in the list as predictions for the adjacencies in the correct solution. Specifically, for each $k = 1, 2, \ldots$, the ROC curve contains a point (x_k, y_k), where x_k is the false positive rate and y_k is the corresponding true positive rate. The ideal ROC curve contains a point $(x_k, y_k) = (0, 1)$, i.e., it is possible to choose k such that the first k (predicted) adjacencies correspond to the true solution.

Figure 3 represents, for each redundancy value, an *average* ROC curve; i.e., each point represents average true and false positives over the first k adjacencies from 10 distinct problem instances.

Our experimental results support the usefulness of the new approach presented in this paper. We conclude with a few specific observations:

- At redundancy of 5x (20 contigs), the first four edges are determined correctly in each of the 10 problem instances. In fact, in 7 out of the 10 problem instances the best-scoring solution is the fully correct permutation (data not shown here).
- If one considers the first 9 edges returned for each of the 10 instances, there are only 4 errors. The other $90 - 4 = 86$ edges are correct.
- As expected, the result of our method improves with increasing redundancy/decreasing number of contigs. At redundancy level 6.5x, all correct edges are predicted perfectly for each of the 10 problem instances.

Note that our simulations were performed on a relatively large target ($>$ 300kb). For the much smaller targets that also arise in practice, fewer restriction sites, and hence, even better performance of our method can be expected.

Due to the efficient algorithms for the two subproblems, we can afford to perform a large number of steps (50,000 in our examples) in the outer loop of our algorithm in Figure 2. We have empirical evidence that such a large number of steps is essential for finding good solutions. The running time of our implementation for 50,000 iterations on a current Sun workstation is on the order of 10 minutes.

Acknowledgements

We thank Nili Avidan (the Weizmann Institute, Rehovot) and Juliane Ramser (Max Planck Institute for Molecular Genetics, Berlin) for providing us with data from real sequencing projects and performing experiments that motivated our modeling decisions. We thank Roded Sharan (Tel Aviv University) for many comments on the manuscript. RS would like to thank Eric Lander (Whitehead Institute, Boston) for an early discussion on the problem and for coining the term RSP. RMK and RS were supported in part by a grant from the US-Israel Binational Science Foundation (BSF). BS was supported in part by the German Academic Exchange Service (DAAD).

7. REFERENCES

[1] W.-W. Cai, R. Chen, R. A. Gibbs, and A. Bradley. A clone-array pooled shotgun strategy for sequencing large genomes. *Genome Research*, 11(10):1619–1623, 2001.

[2] D. Gordon, C. Desmarais, and P. Green. Automated finishing with autofinish. *Genome Research*, 11:614–625, 2000.

[3] R. Karp and S.-Y. R. Li. Two special cases of the assignment problem. *Discrete Mathematics*, 13:129–142, 1975.

[4] S. Kirkpatrick, C. Gelatt, Jr., and M. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, May 1983.

[5] E. Lander et al. Initial sequencing and analysis of the human genome. international human genome sequencing consortium. *Nature*, 409:860–921, 2001.

[6] J. C. Roach, C. Boysen, K. Wang, and H. L. Pairwise end sequencing: a unified approach to genomic mapping and sequencing. *Genomics*, 26(2):345–353, 1995.

[7] J. Swets. *Measuring the Accuracy of Diagnostic Systems*. Academic Press, 1982.

[8] 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(3):500–507, 1999.

[9] J. Venter et al. The sequence of the human genome. 291:1304–1351, 2001.

[10] J. C. Venter, H. O. Smith, and L. Hood. A new strategy for genome sequencing. *Nature*, 381:364, 1996.

[11] J. L. Weber and E. W. Myers. Human whole-genome shotgun sequencing. *Genome Research*, 7(5):401–409, 1997.

Appendix A: NP-Hardness of the RSP Problem

THEOREM 1. *RSP is NP-hard, even if only one enzyme is used, no gaps are allowed and fragment multiplicity is known.*

PROOF. Reduction from Partition: Given integers $a_1, \ldots, a_n$ with $\sum_i a_i = 2T$, we need to determine if there is a subset $S \subset \{1, \ldots, n\}$ such that $\sum_{i \in S} a_i = T$. We construct RSP data for one enzyme with measured fragments T, T. There are n contigs of sizes $a_1, \ldots, a_n$ with no enzyme cut sites in them. Clearly, the reduction is polynomial.

Suppose there is a partition into equal sets. Then putting the fragments of S first, in any order (and orientation), followed by the remaining fragments, and having the cut site at T we get a contiguration with score zero.

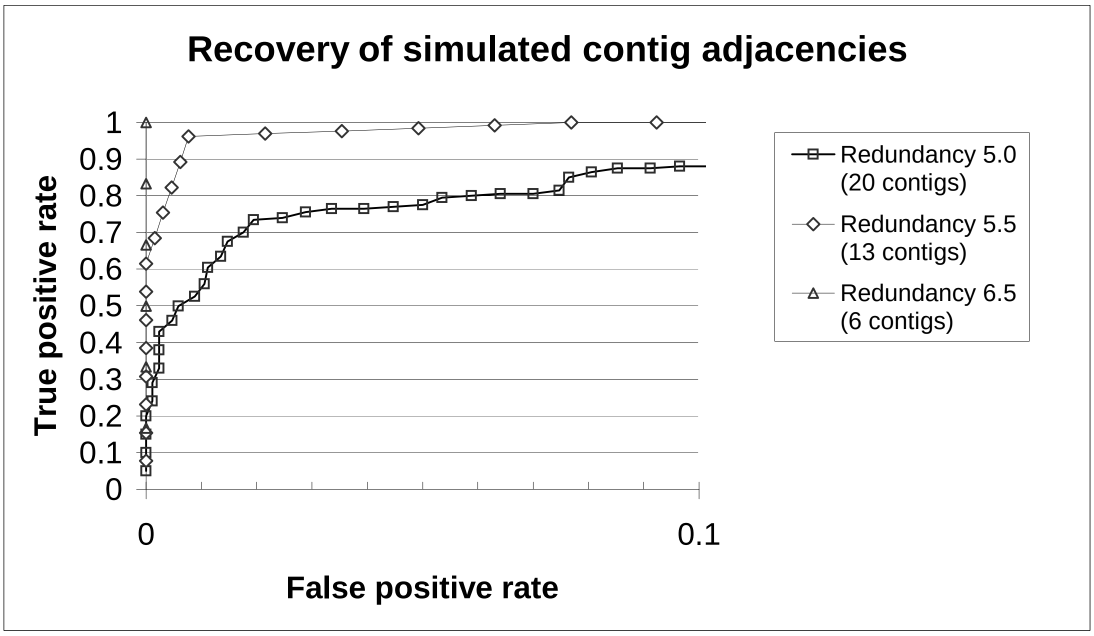

Figure 3: Average predictive power of our algorithm on simulated problem instances

Conversely, if there is a contiguration with score zero then there is a set S of contigs whose sum is T, so S solves the partition problem. $\square$

A similar, strongly polynomial reduction can be made from 3-Partition, showing that the problem is in fact strongly NP-hard. Note that the proof can be modified to avoid a cut site in a (zero size) gap by a slight modification of the data: Pick a sufficiently small $\epsilon > 0$, add an artificial contig of size 2ϵ whose computed fragments are (ϵ, ϵ), and modify the measured fragments to $\{T + \epsilon, T + \epsilon\}$. An alternative modification is to add just one cut site at the very end of one arbitrary contig. Contigs without cut sites can also be avoided by adding a second enzyme and introducing artificial sites at distance δ (for sufficiently small $\delta > 0$) from each contig endpoint, adding $n - 1$ artificial contigs of size 2δ, and having measured fragments $\{a_i - 2\delta | i = 1, \ldots, n\} \cup \{b_i = 2\delta | i = 1, \ldots, n - 1\}$ for the second enzyme.

Appendix B: Reducing RSP matching to linear-cost matching

Formally, for each problem instance P of the RSP matching problem (that allows an optional extra boy at 300bp, and optional inclusion of girls < 300 bp), let us create a problem instance P' of the linear-cost matching problem (without the above extra options), as can be solved by the linear-time algorithm.

- For each girl g, add to P an extra girl/boy pair $(b'(g), g'(g))$ < 300 bp - for simplicity we will call persons < 300 bp "small"),
- Add to P an extra girl/boy pair (b'_0, g'_0) of size 300 bp.

For simplicity, we will call the added girls and boys "imaginary", and the others "real". The following lemma establishes a correspondence between optimal solutions for P and optimal solutions for P'.

LEMMA 1. *Any optimal solution M of P can be converted to an optimal solution M' of P', of the same cost, and vice versa.*

PROOF. Let M be an optimal solution of P. From M we construct the problem instance P', as described above. We obtain a solution to P' by adding an edge $b'(g) - g'(g)$ for each imaginary girl/boy pair (b', g'), as well as an edge $b'(g) - g$ for each small real girl g that is not included in M. It is easy to verify that M' is a solution to P'. Since each added edge has a cost of 0, M' has the same cost as M.

Conversely, let M' be an optimal solution of P'. If there is a real boy b who has an edge to an imaginary girl g', we can modify P' by the following augmentation step. We construct a path that starts with the edge $b - g'$.

Now append g''s imaginary counterpart boy $b'(g')$, and continue with one of b''s girl partners g''. Note that g'' cannot be greater than g', since otherwise the edges $b - g'$ and $b' - g''$ could be rearranged to the cheaper pair $b - g''$ and $b' - g'$, which would contradict the optimality of M'. If g'' is real, stop. Otherwise, iterate the above step with g''. The procedure produces a finite decreasing sequence $(b, g', b'(g'), g'', ...)$ is that must eventually terminate with a real girl g. Now replace the edges $b - g'$, $b' - g''$, ... in M' with the edges $b' - g'$, $b'' - g''$, ..., $b - g$. Observe that the new edge set is still a valid solution of P'. Since the sequence $(b, g', b'(g'), g'', b'(g'')..., g)$ is decreasing, the replacement does not change the cost of the solution. Furthermore, the number of edges between real boys and imaginary girls was decreased by one.

We use the above augmentation step iteratively to eliminate all edges between real boys and imaginary girls without changing the cost of the solution. Finally, we convert the resulting solution of P^* to a solution M of P by the following two steps:

(1) Replace any edge between a large real girl g and a small imagi-

nary boy b' by an edge between g and the imaginary boy b'_0 at 300 bp.

(2) Delete all imaginary boys and girls, except for the imaginary boy b_0, if there is an edge between that boy and one or more large girls.

The resulting graph contains no imaginary persons, with the possible exception of the imaginary boy b_0, and represents the desired solution M of P. Since steps (1) and (2) cannot increase the cost of the edge set, the cost of M is at most as high as the cost of M'.

Finally, observe that neither translation between M and M' has increased the cost, and thus, optimality is preserved in both directions. $\square$

Appendix C: An Example of the Linear-Cost Matching Algorithm

We illustrate the linear-cost matching algorithm with an example. The following figure shows a set of boys and girls on the line, together with their levels.

```
 g       g   b     b       b    g   b       b     g    g  g      g     b g
 0      -1  -1     0       1    1   1       2     2    1  0     -1    -1-1
```

The next figure shows the boys and girls separated by level.

```
level -1
        g   b                                                    g     b g
level 0
 g                b                                       g
level 1
                           b    g   b                   g
level 2
                                             b     g
```

The next figure shows the optimal matching in each level.

```
[h]
level -1
        g   b                                                    g     b g
        -----                                                          ---
level 0
 g                b                                       g
 ------------------
level 1
                           b    g   b                   g
                           ------   --------------------
level 2
                                             b     g
                                             -------
```

The next figure shows an optimal overall matching. It is the unique order-preserving matching between the set of boys and the set of girls that are included in the level-by-level matchings.

```
 g       g   b     b       b    g   b       b     g    g               b g
 ------------              ------   ---------------                    ---
         ----------                         -----------
```

The final figure shows an optimal assignment after the completion of Stage 2.

```
 g       g   b     b       b    g   b       b     g    g  g      g    b g
 ------------              ------   ---------------                   ---
         ----------                         -----------          -----
                                            ---------------
```