Algorithms for Optimizing Production DNA Sequencing

Eva Czabarka [*] Goran Konjevod [†] Madhav V. Marathe [‡] Allon G. Percus [§]
David C. Torney [¶]

Abstract

We discuss the problem of optimally "finishing" a partially sequenced, reconstructed DNA segment. At first sight, this appears to be computationally hard. We construct a series of increasingly realistic models for the problem and show that all of these can in fact be solved to optimality in polynomial time, with near-optimal solutions available in linear time. Implementation of our algorithms could result in a substantial efficiency gain for automated DNA sequencing.

1 Introduction and summary of results

The Human genome comprises 24 chromosomes. Each chromosome contains a double helix of DNA, and each DNA strand is a polymer made up of nucleotides (or *bases*). Four different kinds of bases, denoted A, C, G and T, occur. Each of the two strands on the double helix is directed, and the paired strands have opposite (or *antiparallel*) orientations. The paired DNA strands exhibit what is known as Watson-Crick *complementary* basepairing [6]: if an A appears at a certain position on one strand, a T appears opposite it on the other strand, and vice versa. Likewise, if a C appears at a certain position on one strand, a G appears opposite it on the other strand, and vice versa. (Thus, each of the two paired strands contains essentially the same information).

Sequencing a DNA strand means determining the order in which the bases occur on the strand. The goal of the Human Genome Project is to find a sequence of a representative genome, containing 3×10^9 bases. The most powerful DNA-sequencing procedure is the *chain termination* method developed by Sanger et al. [5, 2]. Given a DNA segment, the Sanger process sequences approximately the first 500 bases (one *read length*) occurring in the segment. Because of the aforementioned particulars of duplex DNA strands, this process, in fact, allows us to sequence one read length from one strand at one end of the segment, and one read length from the other strand at the other end of the segment.

Since only the ends of a DNA segment can be sequenced, it is necessary to obtain many small segments, and then reconstruct the larger sequence from the smaller components. A common method for doing so involves first generating large numbers of *BAC clones* from the chromosomal DNA. These clones are of the order of 10^5 bases in length [2] and are (unspecified) intervals of the double stranded DNA. Through a procedure known as *mapping*, one verifies that the clones one obtains do indeed cover the entire chromosome — generally with some, but minimal, overlap. Each clone, however, is still of the order of hundreds of read lengths, so it must be broken up further. Through hydrodynamic shearing of many copies of an individual clone, a large number of *subclones* are created. To a good approximation, this is a stochastic process, where the subclones are randomly selected intervals from the parent clone, with a uniform distribution over the clone. The lengths of the subclones (if not the locations) are under good experimental control, and are typically of the order of 4 or 5 read lengths. It is these subclones that are then sequenced, from both ends, as described above. This procedure is referred to as *shotgun sequencing* [5, 2], and is depicted schematically in Figure 1a.[1]

The locations of the sequenced regions on the parent clone are not known *a priori*; assembling the sequence requires inferring these locations using computational procedures that take advantage of any over-

[*]Department of Mathematics, University of South Carolina, Columbia, SC 29208. E-mail: `czabarka@math.sc.edu`. This research was supported in part by NSF contract DMS 9701211.

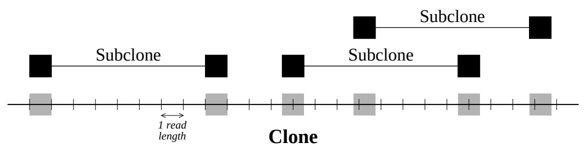
[†]Mathematical Sciences Department, Carnegie Mellon University, Wean Hall, Pittsburgh, PA 15213 and TRANSIMS, MS–M997, Los Alamos National Laboratory, Los Alamos, NM 87545. E-mail: `konjevod@andrew.cmu.edu`.

[‡]Computer Research and Applications Group (CIC–3), MS–P990, Los Alamos National Laboratory, Los Alamos, NM 87545. E-mail: `marathe@lanl.gov`.

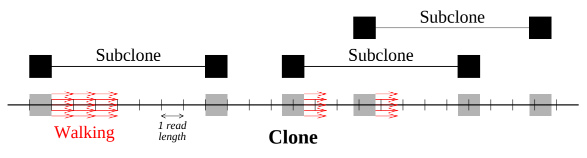
[§]Computer Research and Applications Group (CIC–3), MS–B265, Los Alamos National Laboratory, Los Alamos, NM 87545. E-mail: `percus@lanl.gov`.

[¶]Theoretical Biology and Biophysics Group (T–10) and Center for Human Genome Studies, MS–K710, Los Alamos National Laboratory, Los Alamos, NM 87545. E-mail: `dct@lanl.gov`.

[1]Note that it is sometimes possible to generate subclones directly from the whole genome, avoiding the use of BAC clones and the mapping problem. Our analysis applies equally well to such a case.



(a)



(b)

Figure 1: Schematic depiction of (a) shotgun sequencing, and (b) walking. Tick marks represent read lengths, and lightly shaded regions show shotgun-sequenced intervals on the clone. Walking extends known sequence by one read length at a time.

laps present [1]. Shotgun sequencing therefore involves a necessary redundancy, since enough subclones must be sequenced to insure a reasonable amount of overlap. Furthermore, there is a small error rate inherent in the experiment — around 1%. An overall error rate of one base in 10,000 is desired. Sequencing each base three times, and using "majority rule" would achieve this objective, under the hypothesis of independent errors. Detailed criteria for DNA sequence accuracy were established at a recent meeting at Bermuda [4].

Any moderate amount of shotgun sequencing will presumably leave *some* regions of a parent clone insufficiently sequenced according to these criteria. Once the locations of these regions have been inferred and sequence assembly has taken place insofar as possible — though there is no general means of determining the length of completely unsequenced regions — the sequence gaps require further "finishing" (or "closure"). This is done by *walking* experiments: a primer is constructed using a sequence that has already been determined, allowing a sequenced region to be extended by one read length. This is shown in Figure 1b. Unlike shotgun sequencing, the locations for walking are under good experimental control (apart from some restrictions that, for the sake of clarity, we omit for the moment). The procedure, however, is also much more labor-intensive and time-consuming.

This presents the experimenter with two optimization problems. The first is deciding the right balance of how much (inexpensive, but stochastic) shotgun sequencing to perform, versus how much (expensive, but deterministic) walking to perform, in order to achieve finished sequence according to the established criteria. The second is, given a certain amount of shotgun sequencing, how precisely to perform the walks. Since shotgun subclone positions are random, the insufficiently sequenced regions are generally of a non-integer number of read lengths. Walking experiments, however, sequence one read length at a time. It is therefore a nontrivial problem to decide *where* to place the walks, in order to meet the criteria with minimal redundancy.

Solving these two problems is of more than just theoretical interest: increasing throughput and reducing cost are essential to the ambitious timetable of the Human Genome Project. An efficiency gain of, say, 25% at this stage would considerably ease one of the major bottlenecks in the sequencing process. Our goal in this paper is to address these issues, using a series of progressively more realistic models. We discuss the question of how much shotgun sequencing should be performed, using a relatively simple analytical framework known as the *site model*. We then turn to the algorithmic issue of determining how walking should be done, using a more complex *coverage model*, and find provably optimal strategies that run in polynomial time. We also provide extremely simple and efficient polynomial time approximation algorithms with "good" performance guarantees; these algorithms could be used instead of polynomial time algorithms when computational time is a potential bottleneck. The only other work of similar nature we are aware of is by Gordon, Abajian and Green [3]. In contrast to our algorithms, the work of [3] on algorithms for "automating finishing" do not offer any performance guarantees.

The rest of the paper is organized as follows. In Section 2 we discuss the site model; Section 3 discusses the coverage model and the associated algorithms. Finally Section 4 contain concluding remarks and directions for future work.

2 Site Model

The site model sacrifices accuracy in its portrayal of sequencing experiments, but provides valuable insights. It allows us, notably, to determine the number of shotgun subclones minimizing the average cost for finishing the sequence of a parent clone. This model does not deal with the sequence assembly process itself (thus ignoring the issue of shotgun redundancy necessary to infer the locations of the sequenced regions), and it does not yield finishing algorithms. Nevertheless, its predictions are useful in characterizing the performance of such algorithms. The purpose of this "toy model" is to simplify the process so that standard optimization techniques suffice. The finishing of real sequences will employ more complex combinatorial optimization algorithms.

Imagine a clone as a chain of "sites", each being one read length (say 500 bases) long. Take the length of the entire parent clone to be L_C sites. Now make an important simplification: let all read lengths correspond exactly to these sites, so that everything involves an integer number of sites. We are therefore approximating our system with a coarse one-dimensional lattice. In the schematic representation of Figure 1, all sequenced regions would now lie exactly between two tick marks. This is a tremendous assumption, as it turns the problem of *where* to perform walks into a trivial one. However, it also produces analytical estimates that we could not otherwise obtain. Within the site constraint, we shall now present three different flavors of the model, increasing both in complexity and in closeness to experimental reality.

2.1 Simple case

In the first and most basic flavor of the model, take the cost of shotgun sequencing to be C_S for each read length sequenced, and the cost of walking to be C_W for each read length sequenced (assume $C_W > C_S$). Given n shotgun-sequenced read lengths (which corresponds to $n/2$ subclones, since both ends of a subclone are sequenced), the total shotgun cost will be nC_S. The question is then how much walking is, on average, necessary to fulfill the Bermuda criteria given n shotgun-sequenced sites in random positions.

Due to various experimental considerations, of the L_C sites on the parent clone only the last L_I (a slightly smaller number) are actually of interest in sequencing — this is a region known as the *insert*. Each insert site that is shotgun-sequenced fewer than three times will have to have its deficit made up by walking. The expected total cost $\mathcal{C}_{\text{simple}}$ for shotgun sequencing and subsequent finishing of all insert sites may thus be given as follows:

$$\mathcal{C}_{\text{simple}} = nC_S + \sum_{j=0}^{2} \left\{ (3-j)\, C_W \times \mathbb{E}(\#\text{ insert sites shotgun-sequenced } j \text{ times}) \right\}.$$

The probability of a site being sequenced by any given shotgun subclone is $2/L_C$. Therefore, the expected number of insert sites that are shotgun-sequenced exactly j times is given by a binomial distribution, and

$$\text{(2.1)}\quad \mathcal{C}_{\text{simple}} = nC_S + \sum_{j=0}^{2} \left\{ (3-j)\, C_W \times L_I \binom{n/2}{j} \left(\frac{2}{L_C}\right)^{j} \left(1 - \frac{2}{L_C}\right)^{n/2-j} \right\}.$$

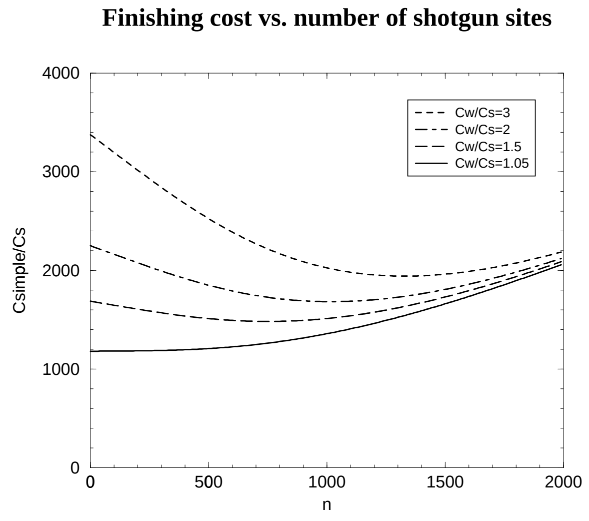


Figure 2: Expected total finishing cost $\mathcal{C}_{\text{simple}}$, in units of C_S, as a function of number of shotgun-sequenced sites n. 4 different ratios C_W/C_S are used. $L_I = 375$ and $L_C = 400$.

Using realistic length values $L_I = 375$ and $L_C = 400$, Figure 2 shows $\mathcal{C}_{\text{simple}}$ as a function of n, given various ratios C_W/C_S. While one cannot in general find a closed-form expression for the $\hat{n}$ that minimizes $\mathcal{C}_{\text{simple}}$ in (2.1), from the figure there appears to be a unique optimum unless C_W/C_S is sufficiently close to unity. One may in fact show analytically that for large L_C, $\hat{n} > 0$ whenever $C_W/C_S > L_C/L_I$.

2.2 General case

The second flavor of our "toy model" takes into account an experimental fact that was ignored earlier: when a walk is performed multiple times at a single site, only the first walk costs the full C_W. All subsequent walks at that site may take advantage of primer material already prepared for the first one, and have a marginal cost that is in fact very close to C_S.

For the site model as described above, this does not much change (2.1). A similar binomial distribution is used, and now:

$$\text{(2.2)}\quad \mathcal{C}_{\text{general}} = nC_S + \sum_{j=0}^{2} \left\{ [C_W + (2-j)\, C_S] \times L_I \binom{n/2}{j} \left(\frac{2}{L_C}\right)^{j} \left(1 - \frac{2}{L_C}\right)^{n/2-j} \right\}.$$

The minima for various ratios C_W/C_S are seen in Figure 3. Note that for a given value of the ratio, walking is generally "cheaper" here than in the simpler

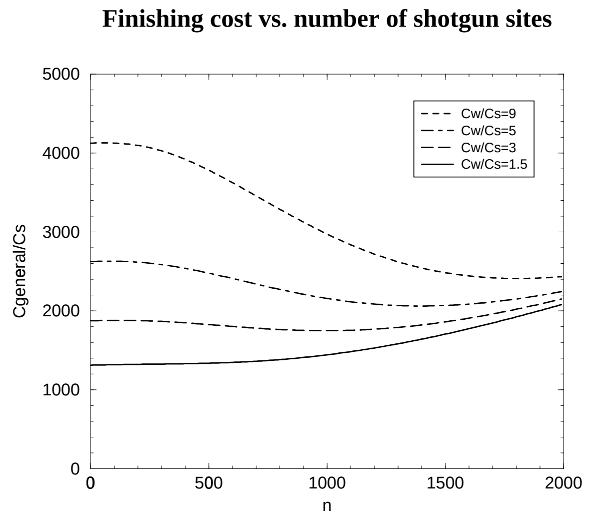


Figure 3: Expected total finishing cost $\mathcal{C}_{\text{general}}$, in units of C_S, as a function of number of shotgun-sequenced sites n. 4 different ratios C_W/C_S are used. $L_I = 375$ and $L_C = 400$.

flavor of the model. Therefore, only for larger values of C_W/C_S does a unique optimum $\hat{n} > 0$ appear.

2.3 Strand case

The final flavor of the site model incorporates a far greater subtlety. We have seen that a shotgun subclone provides one read length of known sequence, from one strand at one end of the subclone, and a second read length of known sequence, from the other strand at the opposite end of the subclone. We have been assuming that knowledge of one DNA strand is equivalent to knowledge of the other strand, as the two exhibit complementary basepairing. This is not completely accurate. The shotgun method can, for example, occasionally insert or delete a base. Given such experimental realities, the Bermuda criteria require not only that each base be sequenced at least three times, but also that *both* strands be represented in the redundant sequencing. Thus, even if a base is shotgun-sequenced three times, if the three trials all fall on the same strand, it must still be sequenced once on the opposite strand.

The strand distinction also complicates the walking cost. As before, the first walk costs C_W. A subsequent walk on a given strand at a given site, however, costs C_S if and only if (a) a previous walk was performed there, *on the same strand*, and (b) the site is covered by a shotgun subclone (though not necessarily sequenced by a shotgun read length). Otherwise it costs the full C_W.

How do these realities change the model? $\mathcal{C}$ must now take into account several new cases. If a site is not subcloned at all, the walking cost will be the full $3C_W$. If a site is subcloned but not shotgun-sequenced, it will be necessary to walk twice on one strand (the second time taking advantage of the economy of multiple walks), and once on the other strand, at a cost of $2C_W + C_S$. If a site is shotgun-sequenced exactly once, it will be sufficient to walk twice on the opposite strand, at a cost of $C_W + C_S$. If a site is shotgun-sequenced twice, regardless of whether these are on the same or opposite strands it will be necessary to walk once more, at a cost of C_W. Finally, if a site is shotgun-sequenced three or more times but on one strand only, it will still be necessary to walk the opposite strand, at a cost of C_W.

The expected total cost is then:

$$\begin{aligned}\mathcal{C}_{\text{strand}} &= nC_S \\ &\quad + 3C_W\,\mathbb{E}(\#\text{ sites not subcloned}) \\ &\quad + [2C_W + C_S] \\ &\qquad \times \mathbb{E}(\#\text{ sites subcloned, not shotgun-sequenced}) \\ &\quad + [C_W + C_S] \\ &\qquad \times \mathbb{E}(\#\text{ sites shotgun-sequenced exactly once}) \\ &\quad + C_W \\ &\qquad \times \mathbb{E}(\#\text{ sites shotgun-sequenced exactly twice}) \\ &\quad + C_W \\ &\qquad \times \mathbb{E}(\#\text{ sites shotgun-sequenced three or more} \\ &\qquad\qquad \text{times on one strand only}).\end{aligned}$$

Since we only need to walk insert sites, and assuming that all subclones have equal length $L_S > 1$, the relevant contributions are

$$\mathbb{E}(\#\text{ sites not subcloned}) = L_I\left(1 - \frac{L_S}{L_C}\right)^{n/2},$$

$$\mathbb{E}(\#\text{ sites subcloned, not shotgun-sequenced}) = L_I\left\{\left(1-\frac{2}{L_C}\right)^{n/2} - \left(1-\frac{L_S}{L_C}\right)^{n/2}\right\},$$

$$\mathbb{E}(\#\text{ sites shotgun-sequenced exactly once}) = L_I\,\frac{n}{2}\,\frac{2}{L_C}\left(1-\frac{2}{L_C}\right)^{n/2-1},$$

$$\mathbb{E}(\#\text{ sites shotgun-sequenced exactly twice}) = L_I\binom{n/2}{2}\left(\frac{2}{L_C}\right)^2\left(1-\frac{2}{L_C}\right)^{n/2-2},$$

$\mathbb{E}(\#$ sites shotgun-sequenced three or more times on one strand only)

$$= 2L_I \left\{ \left(1 - \frac{1}{L_C}\right)^{n/2} - \sum_{j=0}^{2} \binom{n/2}{j} \frac{1}{{L_C}^j} \left(1 - \frac{2}{L_C}\right)^{n/2-j} \right\}.$$

Note that the final expression simply consists of the number of sites not sequenced at all on a given strand, minus the number of sites not sequenced at all on that strand but sequenced exactly j times ($0 \le j \le 2$) on the other strand. The entire quantity is multiplied by two to account for both strands. Collecting the foregoing terms yields

$$(2.3) \quad \mathcal{C}_{\text{strand}} = nC_S + L_I \left\{ (C_W - C_S)\left(1 - \frac{L_S}{L_C}\right)^{n/2} + 2C_W \left(1 - \frac{1}{L_C}\right)^{n/2} + C_S \left(1 - \frac{2}{L_C}\right)^{n/2} + nC_S \frac{1}{L_C}\left(1 - \frac{2}{L_C}\right)^{n/2-1} + 2C_W \binom{n/2}{2} \frac{1}{{L_C}^2}\left(1 - \frac{2}{L_C}\right)^{n/2-2} \right\}.$$

The cost now has a dependence on the subclone length L_S, in addition to the other parameters. As we see in Figure 4, however, this dependence is relatively weak; even varying the value of L_S from 2 to 16 read lengths does not change results dramatically. Note also that for a given ratio C_W/C_S, the total cost $\mathcal{C}_{\text{strand}}$ is significantly greater than $\mathcal{C}_{\text{general}}$. This is due to the additional walks imposed by the strand consideration, as well as the fact that multiple walks are not economical as often. In fact, $\mathcal{C}_{\text{strand}}$ is remarkably close to $\mathcal{C}_{\text{simple}}$ (Figure 2), both in the shape of the curve and in the quantitative cost. Like Figure 2, Figure 4 shows a unique $\hat{n} > 0$ as long as C_W is not too close to C_S. Furthermore, $\hat{n}$ appears very close in the two flavors of the model, suggesting that $\mathcal{C}_{\text{simple}}$ itself provides good estimates of the optimal amount of shotgun sequencing to perform for the strand case.

3 Coverage model

So far, we have addressed the issue of how much shotgun sequencing to perform. In the site model, we restricted the positions of subclones to discrete spacings, of one read length each, on the parent clone. In reality, of course, there is no such constraint; subclones can begin at any base along a DNA strand. Furthermore, while we have examined the expected total cost of shotgun sequencing and finishing in the model, and have seen that there is generally a clear optimal amount of

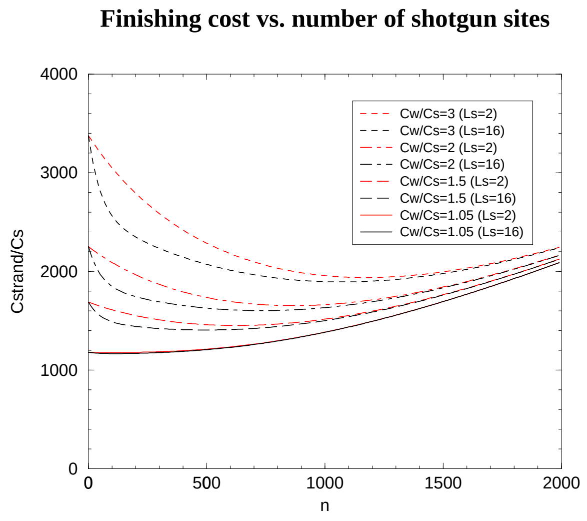


Figure 4: Expected total finishing cost $\mathcal{C}_{\text{strand}}$, in units of C_S, as a function of number of shotgun-sequenced sites n. 4 different ratios C_W/C_S are used; light lines are for subclone length $L_S = 2$ and dark lines for subclone length $L_S = 16$. $L_I = 375$ and $L_C = 400$.

shotgun sequencing in order to minimize this cost, we have not had to pay attention to *how* to perform the walking process optimally. The site constraint makes this a trivial problem.

Without the site constraint, finding an optimal walking algorithm becomes nontrivial. Let us now consider a more realistic model, the *coverage model*, where we no longer require that subclone positions be restricted to discrete sites. For a given layout of shotgun subclones, we now have to "walk" gaps that in general take up a non-integer number of read lengths. This cannot be done with perfect efficiency, *i.e.*, sequencing each base three *and only three* times. The problem is therefore to meet the Bermuda criteria of threefold redundancy while minimizing the excess, or more specifically, minimizing the total walking cost.

One paradigm for visualizing the shotgun coverage profile is the "cityscape" shown in Figure 5. At any given position along the DNA strand, there is a profile height, giving the number of times the base has been sequenced. For the purposes of our analysis, we assume that a walk may be performed starting at any position, and has the effect of incrementing the profile height, by one, over one read length.[2] Walks must be performed so

[2]In reality, experimental constraints somewhat restrict the options of where to perform walks, but for the purposes of a formal understanding of the problem we ignore these for now. We will return to this point in the conclusion.

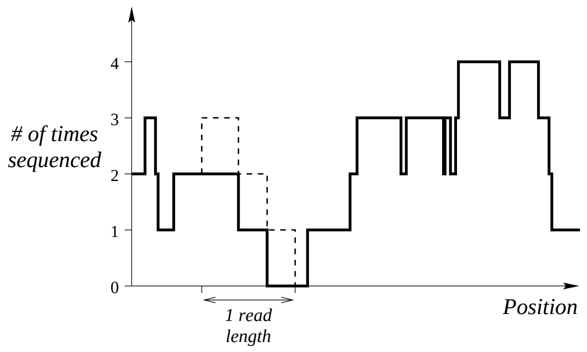


Figure 5: Coverage profile, showing the number of times positions along a section of DNA strand have been shotgun-sequenced. Dashed line shows new profile height after performing one walk.

that the profile ends up with a height of at least three, at all positions. The goal is to determine where to place the walks, so that this can be done at minimal cost.

More formally, the problem may be stated as follows: we are given a set of unit intervals $J_1, \ldots, J_n$ (shotgun-sequenced read lengths) contained inside the region $[0, m_1]$. In the notation of our site model, $m_1 = L_C$. Furthermore, let $m_0 = L_C - L_I$, so that the insert region of interest will be $[m_0, m_1]$.

The set of intervals $\{J_i\}$ are identified with a *profile function* p, defined as:

$$p(x) = |\{J_i \mid x \in J_i\}|.$$

Given a requirement of k-fold redundant coverage ($k = 3$ in the Bermuda criteria), let us call $k - p(x_0)$ the *deficiency* of profile p at a given point x_0. The goal is now to find a new set of unit intervals $W_1, W_2, \ldots, W_\ell$ (walking-sequenced read lengths) allowing us to *cover* the profile over the region $[m_0, m_1]$ at minimal cost. This means that once these new intervals are added to the profile, no more positive deficiencies exist. The cost $\mathcal{C}$ is a function of the new intervals. We will consider the same three cases for $\mathcal{C}$ as we did in the site model.

3.1 Simple case

For the simple flavor of the model, each walk costs a fixed C_W. There is no economy in multiple walking. The cost is thus proportional to the number ℓ of new intervals $W_1, W_2, \ldots, W_\ell$ added. The problem is to decide how to place these intervals so as to minimize the number necessary.

Given required coverage redundancy k, define a *breakpoint* for a profile $p(x)$ to be any point x_0 in the profile such that the deficiency is non-positive immediately to the left of x_0 and positive immediately to the right of x_0. Now consider the following "greedy" walking algorithm:

Algorithm (A)
(1) Let $i = 1$.
(2) Let x_0 be the leftmost breakpoint of p in $[m_0, m_1]$. Place an interval W_i beginning at x_0.
(3) Update profile p to incorporate W_i, i.e.*, increment $p(x)$ on $x \in [x_0, x_0 + 1)$.*
(4) If the profile is not yet covered, increment i and repeat at step (2).

Theorem 3.1. *Algorithm (A) covers the profile with the minimum number of walks.*

Proof. Consider, in left-to-right order, the intervals used in an optimal solution — a solution with a minimum number of walks. Shift each walk as far to the right as possible, whilst preserving the covering. (The final walks shifted in this way may extend past the region of the coverage profile, but this is not a matter of great concern.) Once this procedure is finished, scan the profile again in left-to-right order. By construction, each walk now begins at a point where, without it, there would be a breakpoint. This is precisely the solution produced by Algorithm (A). □

3.2 General case

Let us now reintroduce the distinction between single and multiple walks. As in the general flavor of the site model, let the first walk at a given position cost C_W, but let all subsequent walks at that same position cost only C_S per walk. Thus, we may now have *μ-fold multiple walks*, each costing $C_W + (\mu - 1)C_S$. Defining $\alpha = (C_W - C_S)/C_S$, the cost per walk is then $C_S(\mu + \alpha)$.

The most straightforward way of formulating this version of the problem is to ask for the set of unit intervals $W_1, W_2, \ldots, W_\ell$, with associated multiplicities $\mu_1, \mu_2, \ldots, \mu_\ell$ ($\mu_i \in \{1, \ldots, k\}$), that minimizes

$$\sum_{i=1}^{\ell} (\mu_i + \alpha)$$

subject to the constraint that $\forall x \in [m_0, m_1]$,

$$\sum_{i:\ x \in W_i} \mu_i \geq k - p(x).$$

Clearly, the choice of optimal algorithm now depends on α. Before considering an exact approach, however, let us discuss some possible algorithms giving near-optimal solutions with performance guarantees.

Approximations. Two cases will be examined: where α is small, and where α is large.

Take the case where α is close to 0 (this means that a μ-fold multiple walk is close in cost to μ single walks). Apply Algorithm (A) from the previous subsection.

THEOREM 3.2. *The cost of Algorithm (A) is within a factor* $1 + \alpha(k-1)/(k+\alpha)$ *of optimality.*

Proof. Let ℓ be the number of walks performed in Algorithm (A), weighted by their multiplicities in the case of multiple walks. By Theorem 3.1, there is no way of covering the profile using fewer than the equivalent of ℓ single walks. Each of these costs at least C_S, and in the optimal solution at least one out of every k walks must cost C_W (walks with multiplicity higher than k are never useful), so Opt $\geq \ell[C_S + (C_W - C_S)/k]$.

The most a single walk can cost, however, is C_W, so the cost of Algorithm (A) is less than or equal to ℓC_W, hence less than or equal to Opt $C_W/[C_S + (C_W - C_S)/k]$. The approximation factor is therefore at worst $C_W/[C_S + (C_W - C_S)/k] = 1 + \alpha(k-1)/(k+\alpha)$. □

Now take the case where α is large (this means that μ single walks are substantially more expensive than a μ-fold multiple walk.) Let us consider a new kind of greedy approximation algorithm. We will show that, given a redundancy requirement of $k = 3$ as in the Bermuda criteria, this algorithm provides a performance guarantee of $1 + 2/(\alpha+1)$. Then we will strengthen the analysis slightly, to obtain the guarantee of $1 + 1/(\alpha + 3/2)$ for the same algorithm, under an assumption that is generally satisfied in practice.

ALGORITHM (B)
(1) Let $i = 1$.
(2) Let x_0 *be the leftmost breakpoint of* p *in* $[m_0, m_1]$. *Let* $\mu = \max\{k - p(x) \mid x \in [x_0, x_0+1)\}$. *Place a* μ*-fold interval* W_i *beginning at* x_0.
(3) Update profile p *to incorporate* W_i, i.e., *increase* $p(x)$ *by* μ *on* $x \in [x_0, x_0+1)$.
(4) If the profile is not yet covered, increment i *and repeat at step (2).*

LEMMA 3.3. *Algorithm (B) covers the profile with the minimum number of (potentially multiple) walks.*

Proof. First of all, note the key difference between this and Theorem 3.1. Here, we do not weight the walk count with multiplicities; even if a walk W_i has multiplicity μ_i, it still counts as just one walk for the present purposes. The rest of the proof is similar to that of Theorem 3.1. Consider any solution that uses the minimum number of (potentially multiple) walks, and shift each walk as far to the right as possible. Each walk now begins at a breakpoint, and has a multiplicity corresponding to the most deficient point in the region it spans. Take the first walk (from the left) which is not in the solution provided by Algorithm (B). By construction, Algorithm (B) has a walk starting at the same point, but with higher multiplicity. Increase the multiplicity of the step of the optimal solution to that of the step in Algorithm (B) (this does not increase the number of steps). By repeating this procedure we ultimately obtain the solution produced by Algorithm (B) without increasing the number of steps. □

THEOREM 3.4. *The cost of Algorithm (B) is within a factor* $1 + (k-1)/(\alpha+1)$ *of optimality.*

Proof. Let ℓ be the number of (potentially multiple) walks performed in Algorithm (B). By Lemma 3.3, there is no way of covering the profile using fewer than ℓ such walks, and there is no way that any one of these can cost less than C_W (this is the cost when it has multiplicity 1), so $\ell C_W \leq$ Opt.

The most a multiple walk can cost, however, is $C_W + (k-1)C_S$, so the cost of Algorithm (B) is less than or equal to $\ell[C_W + (k-1)C_S]$, hence less than or equal to Opt$[C_W + (k-1)C_S]/C_W$. The approximation factor is therefore at worst $[C_W + (k-1)C_S]/C_W = 1 + (k-1)/(\alpha+1)$. □

Given $k = 3$ as in the Bermuda criteria, in the case where $\alpha > 1$ ($C_W > 2C_S$) we can strengthen the foregoing analysis slightly and improve the bound. The following lemma will be used.

LEMMA 3.5. *If* $C_W > 2C_S$, *an optimal cover can be created for a profile of height 3 with no three walks overlapping.*

Proof. Let W_1, W_2, W_3 be three intervals in a valid cover, with multiplicities μ_1, μ_2, μ_3, starting at points $x_1 \leq x_2 \leq x_3$ such that $x_3 - x_1 < 1$. If $\mu_1 + \mu_2 \geq 3$, push W_3 to the right. This eliminates the triple overlap while preserving the cover. Otherwise $\mu_1 = \mu_2 = 1$, in which case increase μ_1 and μ_3 by 1, and remove W_2. Since W_2 is contained entirely within the range of W_1 and W_3, this eliminates the triple overlap, preserves the cover, and changes the cost by $2C_S - C_W < 0$. Thus, a triple overlap can always be eliminated without an increase in cost. □

THEOREM 3.6. *For* $k = 3$ *and* $C_W > 2C_S$, *the cost of Algorithm (B) is within a factor* $1 + 1/(\alpha + 3/2)$ *of optimality.*

Proof. Consider an optimal solution. By Lemma 3.5, this can be done in such a way that no three walks ever overlap. Now, working from left to right, eliminate all remaining (two-walk) overlaps by increasing the multiplicity of the first as necessary and shifting the second to the right as necessary. This will increase the solution cost in the following cases:

(a) $\mu_1 = 1$, $\mu_2 = 1$ $\longrightarrow$ $\mu_1 = 2$, $\mu_2 = 1$
(b) $\mu_1 = 1$, $\mu_2 = 2$ $\longrightarrow$ $\mu_1 = 3$, $\mu_2 = 2$
(c) $\mu_1 = 1$, $\mu_2 = 3$ $\longrightarrow$ $\mu_1 = 3$, $\mu_2 = 3$
(d) $\mu_1 = 2$, $\mu_2 = 1$ $\longrightarrow$ $\mu_1 = 3$, $\mu_2 = 1$
(e) $\mu_1 = 2$, $\mu_2 = 2$ $\longrightarrow$ $\mu_1 = 3$, $\mu_2 = 2$
(f) $\mu_1 = 2$, $\mu_2 = 3$ $\longrightarrow$ $\mu_1 = 3$, $\mu_2 = 3$

It may be seen that the worst-case increase in cost occurs in case (b), where the ratio of new to old cost is:

$$\frac{2C_W + 3C_S}{2C_W + 1C_S} = 1 + \frac{1}{\alpha + 3/2}.$$

In this newly modified solution, walks have been shifted to the right so that they no longer overlap. Thus, each walk begins at what would otherwise be a breakpoint. This is precisely the layout of Algorithm (B)'s solutions, though some of the multiplicities might be greater here (there is no way they could be *smaller* and still provide a valid cover). Algorithm (B) is then, at worst, as costly as this procedure, whose cost is at worst a factor of $1 + 1/(\alpha + 3/2)$ over the optimal cost. The approximation factor in Algorithm (B) is therefore at worst $1 + 1/(\alpha + 3/2)$. □

Exact solution. Let us now describe a dynamic programming algorithm that outputs an optimal cover for a profile of height $k = 3$ in time $O(mn^2)$, where m is the insert length $m = m_1 - m_0$ and n is the number of shotgun-sequenced read lengths in the initial profile. We assume hereafter that $C_W > 2C_S$, so that Lemma 3.5 holds.

Dynamic programming algorithms work recursively, optimally solving smaller instances (subproblems) of the problem at hand, and then combining the solutions to these subproblems into an optimal solution for the original problem. In order to describe a dynamic programming algorithm, we therefore need to specify a scheme for decomposing the problem into subproblems, and then establish an upper bound both on the total number of subproblems and on the number of ways of decomposing a subproblem into further subproblems.

The decomposition scheme involves defining the *updated profile*

$$q_i(x) = p(x) + \sum_{j \le i:\ x \in W_j} \mu_j,$$

so that $q_i(x)$ represents the profile at step i, after the ith walk has been added. We now state two useful lemmas:

LEMMA 3.7. *Any optimal solution can be modified at no cost, so that there is a left-to-right ordering of the covering intervals $W_1, \ldots, W_\ell$ in which every W_i begins at the first breakpoint in the profile q_{i-1}.*

Proof. The leftmost interval (W_1) in the optimal solution may be shifted rightwards until its left endpoint lies above the first breakpoint of the profile p, without incurring any additional cost. Now consider the updated profile at step 1, consisting of p with W_1 in its new position, and repeat the same argument for W_2, the next leftmost interval. The lemma follows inductively. □

LEMMA 3.8. *For an optimal solution as described by Lemma 3.7, the number of possible breakpoint positions at which walks $W_1, \ldots, W_\ell$ can begin is at most n per unit length and at most mn over the entire insert length.*

Proof. In the initial profile $p(x)$, there are exactly n shotgun-sequenced read lengths $J_1, \ldots, J_n$, so there can be at most n breakpoints in all. When this profile gets updated with walking intervals, a given breakpoint can move from its initial location x_0 to a new location x_0+1. This means that the set of all possible breakpoint positions corresponds to the initial n positions plus or minus integer values. Any given unit interval $[x_1, x_1+1)$ can contain, at most, n of these, and since the entire insert length is m, there are at most mn in all. □

Henceforth we shall look for an optimal cover that has the form described in Lemma 3.7, with the restrictions imposed by Lemma 3.5. (It is a simple exercise to verify that these two Lemmas are consistent with each other.) Thus, without loss of generality, we restrict ourselves to all feasible solutions where no more than two intervals overlap, and where no interval can be shifted to the right without making the solution infeasible. The following further lemma then holds:

LEMMA 3.9. *In order to determine the optimal placement and multiplicities of all remaining walks $W_{i+1}, \ldots, W_\ell$ necessary to cover an updated profile q_i, it is sufficient to specify:*

(1) The initial profile p.
(2) The position of the first breakpoint x_0 in q_i.
(3) The length by which W_i extends past x_0.
(4) The multiplicity μ_i of W_i.

Proof. From Lemma 3.7, optimal placement of remaining walks only requires knowledge of $q_i(x)$ for $x \ge x_0$, *i.e.*, beyond its first breakpoint. Now, of the i walks already present in q_i, the only one that can possibly extend beyond x_0 is W_i; if, say, W_{i-1} did as well, then subsequently placing W_{i+1} at x_0 (following the prescription of Lemma 3.7) would result in three overlapping walks, violating Lemma 3.5. Thus, to obtain $q_i(x)$ for $x \geq x_0$, we simply increase $p(x)$ by μ_i over $x \in W_i$, all of which information is given. □

Now we present the dynamic programming algorithm itself. Define $\Omega_x^{j,k}$ to be the subproblem of optimally covering an updated profile whose first breakpoint is at the xth possible position (out of the mn allowed by Lemma 3.8), and whose rightmost walk extends j positions past the breakpoint and has multiplicity k. By Lemma 3.9, this information is sufficient to specify the subproblem completely. Then do the following:

Algorithm (C)
(1) Let $x = mn$.
(2) Consider subproblem $\Omega_x^{j,k}$ for all possible j values and all possible k values.
(3) Place a walk at the first breakpoint (position x). All possible multiplicities of this walk (up to 3) give updated profiles corresponding either to a trivial case (for x near mn) or else a subproblem already solved earlier. Compute solutions for all these multiplicities, looking up previously generated solutions as necessary.
(4) Record best solution (best out of the new walk's possible multiplicities) for all j and all k.
(5) Decrement x, and repeat at step (2) until x reaches the initial profile's first breakpoint position.

Theorem 3.10. *Algorithm (C) generates an optimal solution in at most $6mn^2$ operations.*

Proof. From Lemma 3.7, there exists an optimal solution where each walk begins at what would otherwise be a breakpoint. By construction, Algorithm (C) gives the least costly solution of this sort, thus the optimum.

The number of operations performed by Algorithm (C) increases as the total number of lookups per subproblem, times the total number of subproblems. There are at most 3 lookups per subproblem, corresponding to the 3 possible multiplicities for the added walk. The total number of subproblems is equal to the product of the total number of possible values for x, j and k. From Lemma 3.8, there are at most mn possible breakpoint positions for x. Since walks are placed from left to right, the rightmost walk in a subproblem can extend no more than one unit length past the breakpoint, so again from Lemma 3.8 there are at most n possible positions for j. Finally, this rightmost walk cannot have multiplicity greater than 2 unless it does not extend at all past the breakpoint (in which case its multiplicity is anyway unnecessary for specifying the subproblem), so there are at most 2 possible values for k. This gives at most $2mn^2$ possible subproblems. Multiplying this by the 3 possible lookups per subproblem results in at most $6mn^2$ operations. □

3.3 Strand case

We now model the DNA segment — more realistically — as being double-stranded. Recall that for the strand case, both the cost of placing walks and the coverage criteria are more complicated. Walks with multiplicity greater than 1 can only be performed in areas covered by one of the $n/2$ shotgun subclones (Figure 1). We therefore define a new type of profile, the *subclone profile* $s(x)$, to be 1 when x is in a subcloned (though not necessarily sequenced) region and 0 otherwise.

Even though the two strands contain largely the same information, in the double-stranded model walks are always performed on a specific strand. We require that each base be sequenced at least once on each strand, and (as before) at least three times in all. This suggests some refinements to our earlier definitions. The profile $p(x)$ will now consist of two components $p^+(x)$ and $p^-(x)$, collectively denoted by the vector quantity $p^\pm(x)$, representing the shotgun coverage of the two strands. A deficiency will occur at a point x_0 if *either* $p^+(x_0) + p^-(x_0) < 3$ *or* one of the two components of $p^\pm(x)$ equals zero. A breakpoint, as before, is any point where the deficiency becomes positive.

Given these refinements, let us generalize our dynamic programming algorithm to the strand case, producing an optimal cover in time $O(mn^3)$. We start with a weaker form of Lemma 3.7.

Lemma 3.11. *Any optimal double-stranded solution can be modified at no cost, so that there is a left-to-right ordering of the covering intervals $W_1, \ldots, W_\ell$ in which every W_i (with associated multiplicity μ_i) satisfies the following property:*
(1) If $\mu_i = 1$, W_i begins at the first breakpoint x_0 in the updated profile $q_{i-1}^\pm$.
(2) If $\mu_i > 1$, W_i begins at the rightmost point $x_1 \leq x_0$ such that $s(x) = 1$ on $x \in [x_1, x_1 + 1)$.

Proof. Similar to the proof of Lemma 3.7, except that if $\mu_i > 1$, W_i may only be shifted rightwards within the allowable region, *i.e.*, a region covered by shotgun subclones, where $s(x) = 1$. □

Given this ordering of the covering intervals, Lemma 3.8 holds in almost unchanged form: subclone coverage must always end at a breakpoint of $p^\pm$, so the number of possible positions at which the covering intervals W_i can begin is at most n per unit length and mn over the entire insert length (as before).

Lemma 3.9 is revised as follows for the strand case:

Lemma 3.12. *In order to determine the optimal placement and multiplicities of all remaining walks $W_{i+1}, \ldots, W_\ell$ necessary to cover an updated profile $q_i^\pm$, it is sufficient to specify:*

(1) The initial shotgun profile $p^{\pm}$.
(2) The subclone profile s.
(3) The position of the first breakpoint x_0 in $q_i^{\pm}$.
(4) The lengths by which W_{i-1} and W_i extend past x_0.
(5) The strands on which W_{i-1} and W_i are performed.
(5) The multiplicities μ_{i-1} and μ_i.

Proof. From Lemma 3.11, optimal placement of remaining walks only requires knowledge of $q_i^{\pm}(x)$ for $x \geq x_1$, where $x_1 \leq x_0$ is the rightmost point such that $s(x) = 1$ on $x \in [x_1, x_1 + 1)$. If $x_1 < x_0$, the subsequent walk W_{i+1} will be placed at x_0 if $\mu_{i+1} = 1$ and at x_1 if $\mu_{i+1} > 1$; in the latter case, however, it is simple to verify that any following walks will be placed at or after x_0. Therefore, we really only require knowledge of $q_i^{\pm}(x)$ for $x \geq x_0$.

Now, of the i walks already present in $q_i^{\pm}$, only W_{i-1} and W_i can possibly extend past x_0; if W_{i-2} did as well, then we would have three overlapping intervals W_{i-2}, W_{i-1} and W_i, two of which would be on one strand and one of which would be on the other (if they were all on the same strand, the middle interval W_{i-1} could always be moved to the other strand while preserving the cover). However, if two intervals on one strand and one interval on the other strand all extend past x_0, x_0 cannot be a breakpoint.

Thus, to obtain $q_i^{\pm}(x)$ for $x \geq x_0$, we simply update $p^{\pm}(x)$ by μ_{i-1} over $x \in W_{i-1}$ and by μ_i over $x \in W_i$ (for each walk we increase the component of $p^{\pm}$ corresponding to the strand on which the walk is performed), all of which information is given. □

Note that, unlike in the single-strand model, nowhere here do we require that $C_W > 2C_S$.

THEOREM 3.13. *Dynamic programming generates an optimal solution in at most $64mn^3$ operations.*

Proof. The algorithm proceeds very similarly to Algorithm (C), except that now the subproblem must keep track of the position, multiplicity and strand for *two* walks, resulting in $16mn^3$ subproblems. Details are left as an exercise to the reader. Given a subproblem, the next walk may in general be performed on either strand and have either multiplicity 1 or 2 (multiplicity 3 is easily excluded). This gives 4 lookups per subproblem, thus $64mn^3$ operations at most. □

4 Discussion and Conclusions

From relatively simple combinatorial optimization techniques applied in an unconventional arena, we find that polynomial-time algorithms and even good linear-time approximations exist for problems that might at first sight appear computationally hard.

There is a considerable list of further experimental realities that should be incorporated into these models. Optimization subject to probabilistic (rather than known) shotgun positions is one important open problem: sequence assembly is sufficiently difficult that in reality one cannot unambiguously infer the positions of shotgun-sequenced regions. Further constraints on walking positions are also important: experimentally it is not always possible to create a primer at the precise position specified by our algorithms. Other open issues include multiple (variable) finishing criteria, parallelized walking capabilities and double-end clone sequencing.

Nevertheless, the success of dynamic programming in dealing with the restrictions imposed by the double-stranded problem suggests that it is robust enough to yield optimal strategies under additional constraints — with, at worst, a higher exponent in the running time. Analysis along these lines could ultimately lead to complete automation and optimization of sequencing experiments, goals that presently remain distant.

Acknowledgments

The authors wish to acknowledge Mark Mundt and two anonymous SODA reviewers for their valuable input concerning the details of sequencing experiments, and the current state-of-the-art.

References

[1] F. Alizadeh, R.M. Karp, L.A. Newberg and D.K. Weisser, *Physical mapping of chromosomes: a combinatorial problem in mathematical biology*, Algorithmica, 13 (1995), pp. 52–76.

[2] S. Anderson, M.H. de Bruijn, A.R. Coulson, I.C. Eperon, F. Sanger, and I.G. Young, *Complete sequence of bovine mitochondrial DNA. Conserved features of the mammalian mitochondrial genome*, J. Mol. Biol., 156 (1992), pp. 683–717.

[3] D. Gordon, C. Abajian, and P. Green, *Consed: A graphical tool for sequence finishing*, Genome Research, 8 (1998), pp. 195–202.

[4] Human Genome Program, U.S. Department of Energy, *JGI and "Bermuda-quality" sequence*, Human Genome News, 9:3 (1998), p. 7, at: `http://www.ornl.gov/hgmis/publicat/hgn/v9n3/07bermud.html`. See also: `http://www.gene.ucl.ac.uk/hugo/bermuda2.htm`.

[5] F. Sanger, S. Nicklen and A.R. Coulson, *DNA sequencing with chain-terminating inhibitors*, Proc. Nat. Acad. Sci. USA, 74 (1977), pp. 5463–5467.

[6] J.D. Watson and F.H.C. Crick, *A structure for deoxyribose nucleic acid*, Nature, 171 (1953), pp. 737–738.

[7] J.D. Watson, N.H. Hopkins, J.W. Roberts, J.A. Steitz and A.M. Wiener, *Molecular Biology of the Gene*, Fourth Edition, Vol. I, Benjamin/Cummings, Menlo Park, NJ, 1987.