

# Pooled genomic indexing (PGI): mathematical analysis and experiment design

Miklós Csűrös<sup>1,2,3</sup> and Aleksandar Milosavljevic<sup>2,3</sup>

<sup>1</sup> Département d'informatique et de recherche opérationnelle, Université de Montréal, CP 6128 succ. Centre-Ville, Montréal, Québec H3C 3J7, Canada. [csuros@iro.umontreal.ca](mailto:csuros@iro.umontreal.ca)

<sup>2</sup> Human Genome Sequencing Center, Department of Molecular and Human Genetics Baylor College of Medicine

<sup>3</sup> Bioinformatics Research Laboratory, Department of Molecular and Human Genetics, Baylor College of Medicine, Houston, Texas 77030, USA. [amilosav@bcm.tmc.edu](mailto:amilosav@bcm.tmc.edu)

**Abstract.** Pooled Genomic Indexing (PGI) is a novel method for physical mapping of clones onto known macromolecular sequences. PGI is carried out by pooling arrayed clones, generating shotgun sequence reads from pools and by comparing the reads against a reference sequence. If two reads from two different pools match the reference sequence at a close distance, they are both assigned (deconvoluted) to the clone at the intersection of the two pools and the clone is mapped onto the region of the reference sequence between the two matches. A probabilistic model for PGI is developed, and several pooling schemes are designed and analyzed. The probabilistic model and the pooling schemes are validated in simulated experiments where 625 rat BAC clones and 207 mouse BAC clones are mapped onto homologous human sequence.

## 1 Introduction

*Pooled Genomic Indexing* (PGI) is a novel method for physical mapping of clones onto known macromolecular sequences. PGI enables targeted comparative sequencing of homologous regions for the purpose of discovery of genes, gene structure, and conserved regulatory regions through comparative sequence analysis. An application of the basic PGI method to BAC<sup>4</sup> clone mapping is illustrated in Figure 1. PGI first pools arrayed BAC clones, then shotgun sequences the pools at an appropriate coverage, and uses this information to map individual BACs onto homologous sequences of a related organism. Specifically, shotgun reads from the pools provide a set of short (cca. 500 base pair long) random subsequences of the unknown clone sequences (100–200 thousand base pair long). The reads are then individually compared to reference sequences, using standard sequence alignment techniques [1] to find homologies. In a clone-by-clone sequencing strategy [2], the shotgun reads are collected for each clone separately. Because of the pooling in PGI, the individual shotgun reads are not associated with the clones, but detected homologies may be in certain cases. If two reads from two different pools match the reference sequence at a close distance, they are both assigned (deconvoluted) to the clone at the intersection of the two pools. Simultaneously, the clone is mapped onto the region of the reference sequence between the two matches. Subsequently, known genomic or transcribed reference sequences are turned into an index into the yet-to-be sequenced homologous clones across species. As we will see below, this basic pooling scheme is somewhat modified in practice in order to achieve correct and unambiguous mapping.

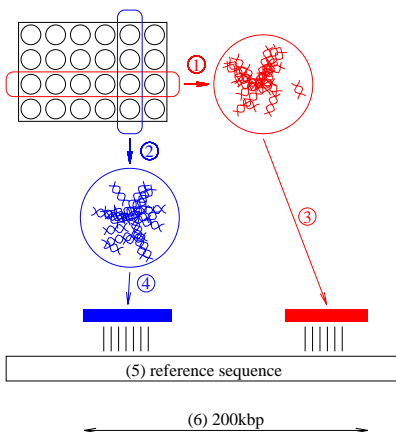
PGI constructs comparative BAC-based physical maps at a fraction (on the order of 1%) of the cost of full genome sequencing. PGI requires only minor changes in the BAC-based sequencing pipeline already established in sequencing laboratories, and thus it takes full advantage of existing economies of scale. The key to the economy of PGI is BAC pooling, which reduces the amount of BAC and shotgun library preparations down to the order of the square root of the number of BAC clones. The depth of shotgun sequencing of the pools is adjusted to fit the evolutionary distance of comparatively mapped organisms. Shotgun sequencing, which represents the bulk of the effort involved in a PGI project, provides useful information irrespective

---

<sup>4</sup> Bacterial Artificial Chromosome

of the pooling scheme. In other words, pooling by itself does not represent a significant overhead, and yet produces a comprehensive and accurate comparative physical map.

Our reason for proposing PGI is motivated by recent advances in sequencing technology [3] that allow shotgun sequencing of BAC pools. The Clone-Array Pooled Shotgun Sequencing (CAPSS) method, described by [3], relies on clone-array pooling and shotgun sequencing of the pools. CAPSS detects overlaps between shotgun sequence reads are used by and assembles the overlapping reads into sequence contigs. PGI offers a different use for the shotgun read information obtained in the same laboratory process. PGI compares the reads against another sequence, typically the genomic sequence of a related species for the purpose of comparative physical mapping. CAPSS does not use a reference sequence to deconvolute the pools. Instead, CAPSS deconvolutes by detecting overlaps between reads: a column-pool read and a row-pool read that significantly overlap are deconvoluted to the BAC at the intersection of the row and the column. Despite the clear distinction between PGI and CAPSS, the methods are compatible and, in fact, can be used simultaneously on the same data set. Moreover, the advanced pooling schemes that we present here in the context of PGI are also applicable to and increase performance of CAPSS, indicating that improvements of one method are potentially applicable to the other.



**Fig. 1.** The Pooled Genomic Indexing method maps arrayed clones of one species onto genomic sequence of another (5). Rows (1) and columns (2) are pooled and shotgun sequenced. If one row and one column fragment match the reference sequence (3 and 4 respectively) within a short distance (6), the two fragments are assigned (deconvoluted) to the clone at the intersection of the row and the column. The clone is simultaneously mapped onto the region between the matches, and the reference sequence is said to index the clone.

In what follows, we propose a probabilistic model for the PGI method. We then discuss and analyze different pooling schemes, and propose algorithms for experiment design. Finally, we validate the method in two simulated PGI experiments, involving 207 mouse and 625 rat BACs.

## 2 Probability of successful indexing

In order to study the efficiency of the PGI strategy formally, define the following values. Let  $N$  be the total number of clones on the array, and let  $m$  be the number of clones within a pool. For simplicity's sake, assume that every pool has the same number of clones, that clones within a pool are represented uniformly, and that every clone has the same length  $L$ . Let  $F$  be the total number of random shotgun reads, and let  $\ell$  be the expected length of a read. The *shotgun coverage*  $c$  is defined by  $c = \frac{F\ell}{NL}$ .

Since reads are randomly distributed along the clones, it is not certain that homologies between reference sequences and clones are detected. However, with larger shotgun coverage, this probability increases rapidly. Consider the particular case of detecting homology between a given reference sequence and a clone. A random fragment of length  $\lambda$  from this clone is aligned locally to the reference sequence and if a significant alignment is found, the homology is detected. Such an alignment is called a *hit*. Let  $M(\lambda)$  be the number of positions at which a fragment of length  $\lambda$  can begin and produce a significant alignment. The probability of a hit for a fixed length  $\lambda$  equals  $M(\lambda)$  divided by the total number of possible start positions for the fragment,  $(L - \lambda + 1)$ .

When  $L \gg \ell$ , the expected probability of a random read aligning to the reference sequence equals

$$p_{\text{hit}} = \mathbb{E} \frac{M(\lambda)}{L - \lambda + 1} = \frac{M}{L}, \quad (1)$$

where  $M$  is a shorthand notation for  $\mathbb{E}M(\lambda)$ . The value  $M$  measures the homology between the clone and the reference sequence. We call this value the *effective length* of the (possibly undetected) index between the clone and the reference sequence in question. For typical definitions of significant alignment, such as an identical region of a certain minimal length,  $M(\lambda)$  is a linear function of  $\lambda$ , and thus  $\mathbb{E}M(\lambda) = M(\ell)$ . Example 1: let the reference sequence be a subsequence of length  $h$  of the clone, and define a hit as identity of at least  $o$  base pairs. Then  $M = h + \ell - 2o$ . Example 2: let the reference sequence be the transcribed sequence of total length  $g$  of a gene on the genomic clone, consisting of  $e$  exons, separated by long ( $\gg \ell$ ) introns, and define a hit as identity of at least  $o$  base pairs. Then  $M = g + e(\ell - 2o)$ .

Assuming uniform coverage and a  $m \times m$  array, the number of reads coming from a fixed pool equals  $\frac{cmL}{2\ell}$ . If there is an index between a reference sequence and a clone in the pool, then the number of hits in the pool is distributed binomially with expected value  $\left(\frac{cmL}{2\ell}\right)\left(\frac{p_{\text{hit}}}{m}\right) = \frac{cM}{2\ell}$ . Propositions 1, 2, and 3 rely on the properties of this distribution, using approximation techniques pioneered by [4] in the context of physical mapping.

**Proposition 1.** *Consider an index with effective length  $M$  between a clone and a reference sequence. The probability that the index is detected equals approximately*

$$p_M \approx \left(1 - e^{-c\frac{M}{2\ell}}\right)^2.$$

(Proof in Appendix.)

Thus, by Proposition 1, the probability of false negatives decreases exponentially with the shotgun coverage level. The expected number of hits for a detected index can be calculated similarly.

**Proposition 2.** *The number of hits for a detected index of effective length  $M$  equals approximately*

$$E_M \approx c\frac{M}{\ell} \left(1 - e^{-c\frac{M}{2\ell}}\right)^{-1}.$$

(Proof in Appendix.)

### 3 Pooling designs

#### 3.1 Ambiguous indexes

The success of indexing in the PGI method depends on the possibility of deconvoluting the local alignments. In the simplest case, homology between a clone and a reference sequence is recognized by finding alignments with fragments from one row and one column pool. It may happen, however, that more than one clone are homologous to the same region in a reference sequence (and therefore to each other). This is the case if the clones overlap, contain similar genes, or contain similar repeat sequences. Subsequently, close alignments

	$C_1$	$C_2$
$R_1$	$B_{11}$	$B_{12}$
$R_2$	$B_{21}$	$B_{22}$

**Fig. 2.** Ambiguity caused by overlap or homology between clones. If clones  $B_{11}$ ,  $B_{12}$ ,  $B_{21}$ , and  $B_{22}$  are at the intersections of rows  $R_1$ ,  $R_2$  and columns  $C_1$ ,  $C_2$  as shown, then alignments from the pools for  $R_1$ ,  $R_2$ ,  $C_1$ ,  $C_2$  may originate from homologies in  $B_{11}$  and  $B_{22}$ , or  $B_{12}$  and  $B_{21}$ , or even  $B_{11}$ ,  $B_{12}$ ,  $B_{22}$ , etc.

may be found between a reference sequence and fragments from more than two pools. If, for example, two rows and one column align to the same reference sequence, an index can be created to the *two* clones at the intersections simultaneously. However, alignments from two rows and two columns cannot be deconvoluted conclusively, as illustrated by Figure 2.

A simple clone layout on an array cannot remedy this problem, thus calling for more sophisticated pooling designs. The problem can be alleviated, for instance, by arranging the clones on more than one array, thereby reducing the chance of assigning overlapping clones to the same array. We propose other alternatives. One of them is based on construction of extremal graphs, while another uses reshuffling of the clones.

In addition to the problem of deconvolution, multiple homologies may also lead to incorrect indexing at low coverage levels. Referring again to the example of Figure 2, assume that the clones  $B_{11}$  and  $B_{22}$  contain a particular homology. If the shotgun coverage level is low, it may happen that the only homologies found are from row  $R_1$  and column  $C_2$ . In that case, the clone  $B_{12}$  gets indexed erroneously. Such indexes are *false positives*. The probability of false positive indexing decreases rapidly with the coverage level as shown by the following result.

**Proposition 3.** *Consider an index between a reference sequence and two clones with the same effective length  $M$ . If the two clones are not in the same row or same column, the probability of false positive indexing equals approximately*

$$p_{\text{FP}} \approx 2e^{-c\frac{M}{\ell}} \left(1 - e^{-c\frac{M}{2\ell}}\right)^2. \quad (2)$$

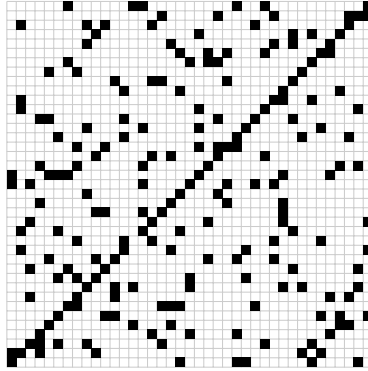
(Proof in Appendix.)

### 3.2 Sparse arrays

The array layout of clones can be represented by an edge-labelled graph  $\mathcal{G}$  in the following manner. Edges in  $\mathcal{G}$  are bijectively labelled with the clones. We call such graphs *clone-labelled*. The pooling is defined by incidence, so that each vertex corresponds to a pool, and the incident edges define the clones in that pool. If  $\mathcal{G}$  is bipartite, then it represents arrayed pooling with rows and columns corresponding to the two sets of vertices, and cells corresponding to edges. Notice that every clone-labelled graph defines a pooling design, even if it is not bipartite. For instance, a clone-labelled full graph with  $N = K(K - 1)/2$  edges defines a pooling that minimizes the number  $K$  of pools and thus number of shotgun libraries, at the expense of increasing ambiguities.

Ambiguities originating from the existence of homologies and overlaps between exactly two clones correspond to cycles of length four in  $\mathcal{G}$ . If  $\mathcal{G}$  is bipartite, and it contains no cycles of length four, then deconvolution is always possible for two clones. Such a graph represents an array in which cells are left empty systematically, so that for all choices of two rows and two columns, at most three out of the four cells at the intersections have clones assigned to them. An array with that property is called a *sparse array*. Figure 3 shows a sparse array. A sparse array is represented by a bipartite graph  $\mathcal{G}^*$  with given size  $N$  and a small number  $K$  of vertices, which has no cycle of length four. This is a specific case of a well-studied problem in extremal graph theory [5] known as the problem of Zarankiewicz. It is known that if  $N > 180K^{3/2}$ , then  $\mathcal{G}^*$  does contain a cycle of length four, hence  $K > N^{2/3}/32$ .

Let  $m$  be a prime power. We design a  $m^2 \times m^2$  sparse array for placing  $N = m^3$  clones, achieving the  $K = \Theta(N^{2/3})$  density, by using an idea of Reiman [6]. The number  $m$  is the size of a pool, i.e., the



**Fig. 3.** Sparse array used in conjunction with the mouse experiments. This  $39 \times 39$  array contains 207 clones, placed at the darkened cells. The array was obtained by randomly adding clones while preserving the sparse property, i.e., the property that for all choices of two rows and two columns, at most three out of the four cells at the intersections have clones assigned to them.

number of clones in a row or a column. Number the rows as  $R_{a,b}$  with  $a, b \in \{0, 1, \dots, m-1\}$ . Similarly, number the columns as  $C_{x,y}$  with  $x, y \in \{0, 1, \dots, m-1\}$ . Place a clone in each cell  $(R_{a,b}, C_{x,y})$  for which  $ax + b = y$ , where the arithmetic is carried out over the finite field  $\mathbb{F}_m$ . This design results in a sparse array by the following reasoning. Considering the affine plane of order  $m$ , rows correspond to lines, and columns correspond to points. A cell contains a clone only if the column's point lies on the row's line. Since there are no two distinct lines going through the same two points, for all choices of two rows and two columns, at least one of the cells at the intersections is empty.

### 3.3 Double shuffling

An alternative to using sparse arrays is to repeat the pooling with the clones reshuffled on an array of the same size. Taking the example of Figure 2, it is unlikely that the clones  $B_{11}$ ,  $B_{12}$ ,  $B_{21}$ , and  $B_{22}$  end up again in the same configuration after the shuffling. Deconvolution is possible if after the shuffling, clone  $B_{11}$  is in cell  $(R'_1, C'_1)$  and  $B_{22}$  is in cell  $(R'_2, C'_2)$ , and alignments with fragments from the pools for  $R_i$ ,  $C_i$ ,  $R'_i$ , and  $C'_i$  are found, except if the clones  $B_{12}$  and  $B_{21}$  got assigned to cells  $(R'_1, C'_2)$  and  $(R'_2, C'_1)$ . Figure 4 shows the possible placements of clones in which the ambiguity is not resolved despite the shuffling. We prove that such situations are avoided with high probability for random shufflings.

Define the following notions. A *rectangle* is formed by the four clones at the intersections of two arbitrary rows and two arbitrary columns. A rectangle is *preserved* after a shuffling if the same four clones are at the intersections of exactly two rows and two columns on the reshuffled array, and the diagonals contain the same two clone pairs as before (see Figure 4).

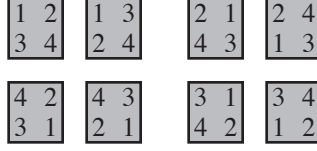
**Theorem 1.** *Let  $R(m)$  be the number of preserved rectangles on a  $m \times m$  array after a random shuffling. Then, for the expected value  $\mathbb{E}R(m)$ ,*

$$\mathbb{E}R(m) = \frac{1}{2} - \frac{2}{m} (1 + o(1)).$$

Moreover, for all  $m > 2$ ,  $\mathbb{E}R(m+1) > \mathbb{E}R(m)$ .

(Proof in Appendix.)

Consequently, the expected number of preserved rectangles equals  $\frac{1}{2}$  asymptotically. Theorem 1 also implies that with significant probability, a random shuffling produces an array with at most one preserved



**Fig. 4.** Possible placements of four clones (1–4) within two rows and two columns forming a rectangle, which give rise to the same ambiguity.

rectangle. Specifically, by Markov’s inequality,  $\mathbb{P}\{R(m) \geq 1\} \leq \mathbb{E}R(m)$ , and thus

$$\mathbb{P}\{R(m) = 0\} \geq \frac{1}{2} + \frac{2}{m}(1 + o(1)).$$

In particular,  $\mathbb{P}\{R(m) = 0\} > \frac{1}{2}$  holds for all  $m > 2$ . Therefore, a random shuffling will preserve no rectangles with at least  $\frac{1}{2}$  probability. This allows us to use a random algorithm: pick a random shuffling, and count the number of preserved rectangles. If that number is greater than zero, repeat the step. The algorithm finishes in at most two steps on average, and takes more than one hundred steps with less than  $10^{-33}$  probability.

*Remark.* Theorem 1 can be extended to non-square arrays without much difficulty. Let  $R(m_r, m_c)$  be the number of preserved rectangles on a  $m_r \times m_c$  array after a random shuffling. Then, for the expected value  $\mathbb{E}R(m_r, m_c)$ ,

$$\mathbb{E}R(m_r, m_c) = \frac{1}{2} - \frac{m_r + m_c}{m_r m_c} (1 + o(1)).$$

Consequently, random shuffling can be used to produce arrays with no preserved rectangles even in case of non-square arrays (such as  $12 \times 8$ , for example).

### 3.4 Pooling designs in general

Let  $\mathcal{B} = \{B_1, B_2, \dots, B_N\}$  be the set of clones, and  $\mathcal{P} = \{P_1, P_2, \dots, P_K\}$  be the set of pools. A general pooling design is described by an incidence structure that is represented by an  $N \times K$  0-1 matrix  $\mathbf{M}$ . The entry  $\mathbf{M}[i, j]$  equals one if clone  $B_i$  is included in  $P_j$ , otherwise it is 0. The *signature*  $\mathbf{c}(B_i)$  of clone  $B_i$  is the  $i$ -th row vector, a binary vector of length  $K$ . In general, the signature of a subset  $\mathcal{S} \subseteq \mathcal{B}$  of clones is the binary vector of length  $K$ , defined by  $\mathbf{c}(\mathcal{S}) = \vee_{B \in \mathcal{S}} \mathbf{c}(B)$ , where  $\vee$  denotes the bitwise OR operation. In order to assign an index to a set of clones, one first calculates the signature  $\mathbf{x}$  of the index defined as a binary vector of length  $K$ , in which the  $j$ -th bit is 1 if and only if there is a hit coming from pool  $P_j$ . For all binary vectors  $\mathbf{x}$  and  $\mathbf{c}$  of length  $K$ , define

$$\Delta(\mathbf{x}, \mathbf{c}) = \begin{cases} \sum_{j=1}^K (\mathbf{c}_j - \mathbf{x}_j) & \text{if } \forall j = 1 \dots K: \mathbf{x}_j \leq \mathbf{c}_j; \\ \infty & \text{otherwise;} \end{cases} \quad (3a)$$

and let

$$\Delta(\mathbf{x}, \mathcal{B}) = \min_{\mathcal{S} \subseteq \mathcal{B}} \Delta(\mathbf{x}, \mathbf{c}(\mathcal{S})). \quad (3b)$$

An index with signature  $\mathbf{x}$  can be deconvoluted unambiguously if and only if the minimum in Equation (3b) is unique. The *weight*  $w(\mathbf{x})$  of a binary vector  $\mathbf{x}$  is the number of coordinates that equal one, i.e.,  $w(\mathbf{x}) = \sum_{j=1}^K \mathbf{x}_j$ . Equation (3a) implies that if  $\Delta(\mathbf{x}, \mathbf{c}) < \infty$ , then  $\Delta(\mathbf{x}, \mathbf{c}) = w(\mathbf{c}) - w(\mathbf{x})$ .

Similar problems to PGI pool design have been considered in other applications of combinatorial group testing [7], and pooling designs have often been used for clone library screening [8]. The design of sparse arrays based on combinatorial geometries is a basic design method in combinatorics (eg., [9]). Reshuffled array designs are sometimes called transversal designs. Instead of preserved rectangles, [10] consider *collinear* clones, i.e., clone pairs in the same row or column, and propose designs with the unique collinearity condition, in which clone pairs are collinear at most once on the reshuffled arrays. Such a condition is more restrictive than ours and leads to incidence structures obtained by more complicated combinatorial methods than our random algorithm. We describe here a construction of arrays satisfying the unique collinearity condition. Let  $q$  be a prime power. Based on results of design theory [9], the following method can be used for producing up to  $q/2$  reshuffled arrays, each of size  $q \times q$ , for pooling  $q^2$  clones. Let  $\mathbb{F}_q$  be a finite field of size  $q$ . Pools are indexed with elements of  $\mathbb{F}_q^2$  as  $P_{x,y} : x, y \in \mathbb{F}_q$ . Define pool set  $\mathcal{P}_i = \{P_{i,y} : y \in \mathbb{F}_q\}$  for all  $i$  and  $\mathcal{P} = \cup_i \mathcal{P}_i$ . Index the clones as  $B_{a,b} : a, b \in \mathbb{F}_q$ . Pool  $P_{x,y}$  contains clone  $B_{a,b}$  if and only if  $y = a + bx$  holds.

**Proposition 4.** *We claim that the pooling design described above has the following properties.*

1. each pool belongs to exactly one pool set;
2. each clone is included in exactly one pool from each pool set;
3. for every pair of pools that are not in the same pool set there is exactly one clone that is included in both pools.

(Proof in Appendix.)

Let  $d \leq q/2$ . This pooling design can be used for arranging the clones on  $d$  reshuffled arrays, each one of size  $q \times q$ . Select  $2d$  pool sets, and pair them arbitrarily. By Properties 1–3 of the design, every pool set pair can define an array layout, by setting one pool set as the row pools and the other pool set as the column pools. Moreover, this set of reshuffled arrays gives a pooling satisfying the unique collinearity condition by Property 3 since two clones are in the same row or column on at most one array.

Similar questions also arise in coding theory, in the context of superimposed codes [11, 12]. Based on the idea of [11], consider the following pooling design method using error-correcting block codes. Let  $\mathcal{C}$  be a code of block length  $n$  over the finite field  $\mathbb{F}_q$ . In other words, let  $\mathcal{C}$  be a set of length- $n$  vectors over  $\mathbb{F}_q$ . A corresponding binary code is constructed by replacing the elements of  $\mathbb{F}_q$  in the codewords with binary vectors of length  $q$ . The substitution uses binary vectors of weight one, i.e., vectors in which exactly one coordinate is 1, using the simple rule that the  $z$ -th element of  $\mathbb{F}_q$  is replaced by a binary vector in which the  $z$ -th coordinate is 1. The resulting binary code  $\mathcal{C}'$  has length  $qn$ , and each binary codeword has weight  $n$ . Using the binary code vectors as clone signatures, the binary code defines a pooling design with  $K = qn$  pools and  $N = |\mathcal{C}|$  clones, for which each clone is included in  $n$  pools. If  $d$  is the minimum distance of the original code  $\mathcal{C}$ , i.e., if two codewords differ in at least  $d$  coordinates, then  $\mathcal{C}'$  has minimum distance  $2d$ . In order to formalize this procedure, define  $\phi$  as the operator of “binary vector substitution,” mapping elements of  $\mathbb{F}_q$  onto column vectors of length  $q$ :  $\phi(0) = [1, 0, \dots, 0]$ ,  $\phi(1) = [0, 1, 0, \dots, 0]$ ,  $\dots$ ,  $\phi(q-1) = [0, \dots, 0, 1]$ . Furthermore, for every codeword  $\mathbf{c}$  represented as a row vector of length  $n$ , let  $\phi(\mathbf{c})$  denote the  $q \times n$  array, in which the  $j$ -th column vector equals  $\phi(\mathbf{c}_j)$  for all  $j$ . Enumerating the entries of  $\phi(\mathbf{c})$  in any fixed order gives a binary vector, giving the signature for the clone corresponding to  $\mathbf{c}$ . Let  $f : \mathbb{F}_q^n \mapsto \mathbb{F}_2^{qn}$  denote the mapping of the original codewords onto binary vectors defined by  $\phi$  and the enumeration of the matrix entries.

*Designs from linear codes* A linear code of dimension  $k$  is defined by a  $k \times n$  generator matrix  $\mathbf{G}$  with entries over  $\mathbb{F}_q$  in the following manner. For each message  $\mathbf{u}$  that is a row vector of length  $k$  over  $\mathbb{F}_q$ , a codeword  $\mathbf{c}$  is generated by calculating  $\mathbf{c} = \mathbf{u}\mathbf{G}$ . The code  $\mathcal{C}_{\mathbf{G}}$  is the set of all codewords obtained in this way. Such a linear code with minimum distance  $d$  is called a  $[n, k, d]$  code. It is assumed that the rows of  $\mathbf{G}$  are linearly independent, and thus the number of codewords equals  $q^k$ . Linear codes can lead to designs with balanced pool sizes as shown by the next lemma.

**Lemma 1.** *Let  $\mathcal{C}$  be a  $[n, k, d]$  code over  $\mathbb{F}_q$  with generator matrix  $\mathbf{G}$ , and let  $f : \mathbb{F}_q^n \mapsto \mathbb{F}_2^{qn}$  denote the mapping of codewords onto binary vectors as defined above. Let  $\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \dots, \mathbf{u}^{(q^k)}$  be the lexicographic enumeration of length  $k$  vectors over  $\mathbb{F}_q$ . For an arbitrary  $1 \leq N \leq q^k$ , let the incidence matrix  $\mathbf{M}$  of the pooling*

be defined by the mapping  $f$  of the first  $N$  codewords  $\{\mathbf{c}^{(i)} = \mathbf{u}^{(i)} \mathbf{G} : i = 1, \dots, N\}$  onto binary vectors. If the last row of  $\mathbf{G}$  has no zero entries, then every column of  $\mathbf{M}$  has  $\lfloor \frac{N}{q} \rfloor$  or  $\lceil \frac{N}{q} \rceil$  ones, i.e., every pool contains about  $m = \frac{N}{q}$  clones.

(Proof in Appendix.)

*Designs from MDS codes* A  $[n, k, d]$  code is *maximum distance separable* (MDS) if it has minimum distance  $d = n - k + 1$ . (The inequality  $d \leq n - k + 1$  holds for all linear codes, so MDS codes achieve maximum distance for fixed code length and dimension, hence the name.) The Reed-Solomon codes are MDS codes over  $\mathbb{F}_q$ , and are defined as follows. Let  $n = q - 1$ , and  $\alpha_0, \alpha_2, \dots, \alpha_{n-1}$  be different non-zero elements of  $\mathbb{F}_q$ . The generator matrix  $\mathbf{G}$  of the RS( $n, k$ ) code is

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ \alpha_0 & \alpha_1 & \dots & \alpha_{n-1} \\ \alpha_0^2 & \alpha_1^2 & \dots & \alpha_{n-1}^2 \\ \dots & \dots & \dots & \dots \\ \alpha_0^{k-1} & \alpha_1^{k-1} & \dots & \alpha_{n-1}^{k-1} \end{pmatrix}.$$

Using a mapping  $f: \mathbb{F}_q^n \mapsto \mathbb{F}_2^{qn}$  as before, for the first  $N \leq q^k$  codewords, a pooling design is obtained with  $N$  clones and  $K$  pools. This design has many advantageous properties. Kautz and Singleton [11] prove that if  $t = \lfloor \frac{n-1}{k-1} \rfloor$ , then the signature of any  $t$ -set of clones is unique. Since  $\alpha_i^{k-1} \neq 0$  in  $\mathbf{G}$ , Lemma 1 applies and thus each pool has about the same size  $m = \frac{N}{q}$ .

**Proposition 5.** *Suppose that the pooling design with  $N = q^k$  is based on a RS( $n, k$ ) code and  $\mathbf{x}$  is a binary vector of positive weight and length  $K$ . If there is a clone  $B$  such that  $\Delta(\mathbf{x}, \mathbf{c}(B)) < \infty$ , then  $\Delta(\mathbf{x}, \mathcal{B}) = n - w(\mathbf{x})$ , and the following holds. If  $w(\mathbf{x}) \geq k$ , then the minimum in Equation (3b) is unique, and is attained for the singleton set containing  $B$ . Conversely, if  $w(\mathbf{x}) < k$ , then the minimum in Equation (3b) is attained for  $q^{k-w(\mathbf{x})}$  choices of singleton clone sets.*

(Proof in Appendix.)

For instance, a design based on the RS(6, 3) code has the following properties.

- 343 clones are pooled in 42 pools;
- each clone is included in 6 pools, if at least 3 of those are included in an index to the clone, the index can be deconvoluted unambiguously;
- each pool contains 49 clones;
- signatures of 2-sets of clones are unique and have weights 10–12;
- signatures of 3-sets of clones have weights between 12 and 18; if the weight of a 3-sets' signature is less than 14, than the signature is unique (determined by a computer program).

The success of indexing in the general case is shown by the following claim.

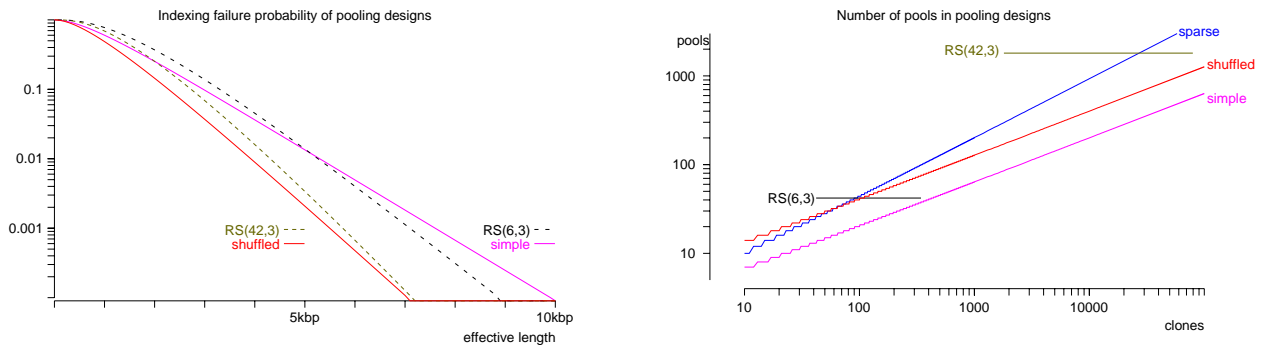
**Proposition 6.** *Consider an index with effective length  $M$  between a clone and a reference sequence. If the clone appears in  $n$  pools, and a set of at least  $n_{\min} \leq n$  of these pools uniquely determines the clone, then the probability that the index is detected equals*

$$p_M = \sum_{t=n_{\min}}^n \binom{n}{t} (1 - p_0)^t p_0^{n-t}$$

where  $p_0 \approx e^{-c \frac{M}{n^2}}$ .

(Proof in Appendix.)

In simple arrayed pooling  $n_{\min} = n = 2$ . In the pooling based on the RS(6, 3) code,  $n_{\min} = 3$ ,  $n = 6$ . When is the probability of success larger with the latter indexing method, at a fixed coverage? The probabilities can be compared using the following analogy. Let 6 balls be colored with red and green independently, each ball is colored randomly to green with probability  $p$ . Let  $\mathcal{X}$  denote the event that at least one of the first three balls, and at least one of the last three balls are red:  $\mathbb{P}\mathcal{X} = (1-p^3)^2$ . Let  $\mathcal{Y}$  denote the event that at least three balls are red:  $\mathbb{P}\mathcal{Y} = \sum_{t=3}^6 \binom{6}{t} (1-p)^t p^{6-t}$ .  $\mathbb{P}\mathcal{X}$  equals the probability of successful indexing in simple arrayed pooling (with  $p_0 = p^3$ ).  $\mathbb{P}\mathcal{Y}$  equals the probability of successful indexing in RS(6,3) pooling (with  $p_0 = p$ ). We claim that  $\mathbb{P}\mathcal{Y} > \mathbb{P}\mathcal{X}$  if  $p < 2/11$ . Consider the event  $\mathcal{X} - \mathcal{Y}$ : when exactly one of the first three balls is red and exactly one of the second three balls is red.  $\mathbb{P}(\mathcal{X} - \mathcal{Y}) = (3(1-p)p^2)^2$ . Consider the event  $\mathcal{Y} - \mathcal{X}$ : when the first three or the second three balls are red, and the others green.  $\mathbb{P}(\mathcal{Y} - \mathcal{X}) = 2(1-p)^3 p^3$ . Now,  $\mathbb{P}\mathcal{Y} > \mathbb{P}\mathcal{X}$  if and only if  $\mathbb{P}(\mathcal{Y} - \mathcal{X}) > \mathbb{P}(\mathcal{X} - \mathcal{Y})$ , i.e., when  $p < 2/11$ . The inequality holds if  $M > \ell \left( -6c \ln(2/11) \right) \approx 10c\ell$ . Thus, for longer homologous regions (cca. 5000 bp effective length if  $c = 1$ ), the RS(6,3) pooling is predicted to have better success. As  $c$  grows, simple arrayed pooling becomes better for the same fixed index. Furthermore, at  $p < 2/11$ , even simple arrayed pooling gives around 99% or higher success probability, thus the improvements are marginal, while the difference between the probabilities is significantly larger when  $p$  is large. On the other hand, a similar argument shows that double shuffling with unique collinearity ( $n = 4$ ,  $n_{\min} = 2$ ) has always higher success probability than simple arrayed pooling. Figure 5 compares the probabilities of successful indexing for some pooling designs.



**Fig. 5.** The graphs on the left-hand side show the probabilities for failing to find an index as a function of the effective length. The graphs are plotted for coverage  $c = 1$  and expected shotgun read length  $\ell = 500$ . The values are calculated from Proposition 6 for simple arraying and double shuffling with unique collinearity, as well as for designs based on the RS(6, 3) and RS(42, 3) codes. Notice that in case of a simple index, the failure probabilities for the sparse array design are the same as for the simple array. The graphs on the right-hand side compare the costs of different pooling designs by plotting how the number of pools depends on the total number of clones in different designs.

## 4 Experiments

We tested the efficiency of the PGI method for indexing mouse and rat clones by human reference sequences in simulated experiments. The reference databases included the public human genome draft sequence [2], the Human Transcript Database (HTDB) [13], and the Unigene database of human transcripts [14]. Local alignments were computed using BLASTN [1] with default search parameters (word size=11, gap open cost=5, gap extension cost=2, mismatch penalty=2). A hit was defined as a local alignment with an E-value less than  $10^{-5}$ , a length of at most 40 bases, and a score of at least 60.

In the case of transcribed reference sequences, hits on the same human reference sequence were grouped together to form the indexes. In the case of genomic sequences, we grouped close hits together within the same reference sequence to form the indexes. In particular, indexes were defined as maximal sets of hits on the same reference sequence, with a certain threshold on the maximum distance between consecutive hits, called the *resolution*. After experimenting with resolution values between 1kbp and 200kbp, we decided to use 2kbp resolution throughout the experiments. A particular difficulty we encountered in the experiments was the abundance of repetitive elements in eukaryotic DNA. If part of a shotgun read has many homologies in the human sequences, as is the case with common repeat sequences, the read generates many hits. Conversely, the same human sequence may be homologous to many reads. Accordingly, repeats in the arrayed clones correspond to highly ambiguous indexes, and human-specific repeats may produce large number of indexes to the same clone. Whereas in the cases of rat and mouse, it is possible to use a database of repeat sequences such as Repbase [15], such information is not available for many other species. We thus resorted to a different technique for filtering out repeats. We simply discarded shotgun reads that generated more than twelve hits, thereby eliminating almost all repeats without using a database of repeated elements.

For the deconvolution, we set the maximum number of clones to three, that is, each index was assigned to one, two, or three clones, or was declared ambiguous.

Finally, due to the fact that pooling was simulated and that in all experiments the original clone for each read was known enabled a straightforward test of accuracy of indexing: an index was considered accurate if both reads deconvoluted to the correct original clone.

	Pools	Number of correctly indexed clones			Number of correct indexes / false positives		
		UG	HTDB	HS	UG	HTDB	HS
simple	29	159 (77%)	139 (67%)	172 (83%)	723 / 248	488 / 108	1472 / 69
RS(6, 3)	42	172 (83%)	-	-	611 / 150	-	-
shuffled	58	172 (83%)	150 (72%)	180 (87%)	756 / 76	514 / 18	1549 / 238
sparse	78	175 (85%)	152 (74%)	185 (88%)	823 / 22	569 / 11	1634 / 69

**Table 1.** Experimental results for simulated indexing of 207 mouse clones with coverage level 2. The table gives the results for four pooling designs (simple, shuffled, sparse array, and RS(6,3)), and three databases of reference sequences: Unigene (UG), HTDB, and human genome draft (HS). The RS(6,3) design was used with the UG database only.

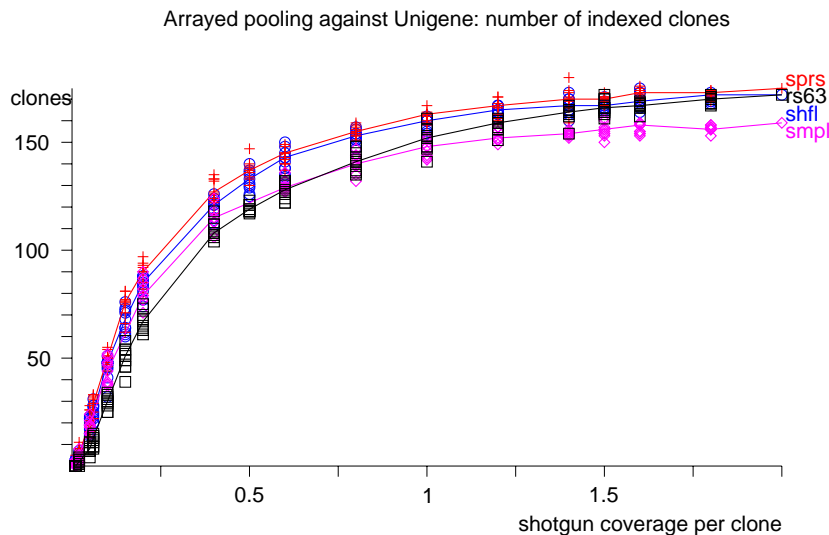
#### 4.1 Mouse experiments

In one set of experiments we studied the efficiency of indexing mouse clones with human sequences. We selected 207 phase 3 sequences in the mouse sequencing project with lengths greater than 50k bases. We used four pooling designs: two  $14 \times 15$  arrays for double shuffling, a  $39 \times 39$  sparse array, shown in Figure 3, and a design based on the RS(6,3) code with 42 pools.

Random shotgun reads were produced by simulation. Each random read from a fixed pool was obtained in the following manner. A clone was selected randomly with probabilities proportional to clone lengths. A read length was picked using a Poisson distribution with mean  $\ell = 550$ , truncated at 1000. The random position of the read was picked uniformly from the possible places for the read length on the selected clone. Each position of the read was corrupted independently with probability 0.01. The procedure was repeated to attain the desired coverage within each pool. Fragments were generated for the different pooling designs with  $c = 2$ .

The results of the experiments are summarized in Table 1. The main finding is that 67%–88% of the clones have at least one correct index, and 500–1600 correct indexes are created, depending on the array design and the reference database. Note that double shuffling and sparse array methods significantly reduce the number of false positives, as expected based on the discussion above. One of the reasons for the excellent

performance of the sparse array method is the fact that BAC redundancy in the data set does not exceed 2X. In the course of the experiments, between 7% and 10% of a total of approximately 121,400 simulated reads gave alignments against human sequences that were informative for the deconvolution, a percentage that is consistent with the expected overall level of genomic sequence conservation between mouse and human.



**Fig. 6.** Number of correctly mapped clones is indicated as a function of shotgun coverage for three different pooling schemes. PGI was tested in a simulated experiment involving 207 publicly available mouse genomic sequences of length between 50kbp and 300kbp. Pooling and shotgun sequencing were then simulated. For each coverage level, reads for  $c = 2$  were resampled ten times. Graphs go through the median values for four pooling designs: simple (smpl), shuffled (shfl), and sparse (sprs), and Reed-Solomon (rs63). Deconvolution was performed using human Unigene database. Notice that the curves level off at approximately  $c = 1.0$ , indicating limited benefit of much greater shotgun coverages.

In order to explore the effect of shotgun coverage on the success of indexing, we repeated the indexing with lower coverage levels. We resampled the simulated reads by selecting appropriate portions randomly from those produced with coverage 2. Figure 6 plots the number of indexed clones as a function of coverage. It is worth noting that even for  $c = 0.5$ , about 2/3 of the clones get indexed by at least one human sequence. In addition, the curves level off at about  $c = 1.0$ , and higher coverage levels yield limited benefits.

## 4.2 Rat experiments

In contrast to the mouse experiments, PGI simulation on rat was performed using publicly available real shotgun reads from individual BACs being sequenced as part of the rat genome sequencing project. The only simulated aspect was BAC pooling, for which we pooled reads from individual BACs computationally. We selected a total of 625 rat BACs, each with more than 570 publicly available reads from the rat sequencing project. An average of 285 random reads per clone were used in each pool — corresponding to an approximate  $c = 1.5$  coverage. The results are summarized in Table 2.

The lower percentage of correctly mapped clones only partially reflects the lower coverage (1.5 for rat vs. 2 for mouse). A much more important factor contributing to the difference is the fact that the mouse sequences used in our experiments are apparently richer in genes and thus contain more conserved regions that contribute to the larger number of BACs correctly mapped onto human sequences.

	correct indexes	false positives	indexed clones
simple	1418	236	384 (61%)
shuffled	1383	30	409 (65%)
sparse	1574	17	451 (72%)

**Table 2.** Experimental results on simulated indexing of 625 rat clones by Unigene sequences with coverage level 1.5.

## 5 Discussion

PGI is a novel method for physical mapping of clones onto known macromolecular sequences. It employs available sequences of humans and model organisms to index genomes of new organisms at a fraction of full genome sequencing cost. The key idea of PGI is the pooled shotgun library construction, which reduces the amount of library preparations down to the order of the square root of the number of BAC clones. In addition to setting priorities for targeted sequencing, PGI has the advantage that the libraries and reads it needs can be reused in the sequencing phase. Consequently, it is ideally suited for a two-staged approach to comparative genome explorations yielding maximum biological information for given amounts of sequencing efforts.

We presented a probabilistic analysis of indexing success, and described pooling designs that increase the efficiency of *in silico* deconvolution of pooled shotgun reads. Using publicly available mouse and rat sequences, we demonstrated the power of the PGI method in simulated experiments. In particular, we showed that using relatively few shotgun reads corresponding to 0.5-2.0 coverage of the clones, 60-90% of the clones can be indexed with human genomic or transcribed sequences.

Due to the low level of chromosomal rearrangements across mammals, the order of BACs in a comparative physical map should provide an almost correct ordering of BACs along the genome of a newly indexed mammal. Such information should be very useful for whole-genome sequencing of such organisms. Moreover, the already assembled reference sequences of model organisms onto which the BACs are mapped may guide the sequence assembly of the homologous sequence of a newly sequenced organism<sup>5</sup> [16].

Comparative physical maps will allow efficient, targeted, cross-species sequencing for the purpose of comparative annotation of genomic regions in model organisms that are of particular biomedical importance. PGI is not limited to the mapping of BAC clones. Other applications currently include the mapping of arrayed cDNA clones onto genomic or known full or partial cDNA sequences within and across species, and the mapping of bacterial genomic clones across different bacterial strains. Sampling efficiency of PGI is in practice increased by an order of magnitude by sequencing short sequence fragments. This is accomplished by breaking the pooled DNA into short fragments, selecting them by size, forming concatamers by ligation, and then sequencing the concatamers [17–19]. Assuming a sequence read of 600bp and tag size of 20–200bp, a total of 3–30 different clones may be sampled in a single sequencing reaction. This technique is particularly useful when the clone sequences and reference sequences are highly similar, e.g., cDNA mapping against the genome of the same species, bacterial genome mapping across similar strains, and mapping of primate genomic BACs against human sequence.

<sup>5</sup> Claim 1 of US Patent 6,001,562 [16] reads as follows: A method for detecting sequence similarity between at least two nucleic acids, comprising the steps of:

- (a) identifying a plurality of putative subsequences from a first nucleic acid;
- (b) comparing said subsequences with at least a second nucleic acid sequence; and
- (c) aligning said subsequences using said second nucleic acid sequence in order to simultaneously maximize
  - (i) matching between said subsequences and said second nucleic acid sequence and
  - (ii) mutual overlap between said subsequences,

whereby said aligning predicts a subsequence that occurs within both said first and said second nucleic acids.

*Acknowledgments* The authors are grateful to Richard Gibbs and George Weinstock for sharing pre-publication information on CAPSS and for useful comments, and to Paul Havlak and David Wheeler for contributing database access and computational resources at HGSC.

## References

1. Altschul, S.F., Madden, T.L., Schäffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J.: Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* **25** (1997) 3389–3402
2. IHGSC: Initial sequencing and analysis of the human genome. *Nature* **609** (2001) 860–921
3. Cai, W.W., Chen, R., Gibbs, R.A., Bradley, A.: A clone-array pooled strategy for sequencing large genomes. *Genome Res.* **11** (2001) 1619–1623
4. Lander, E.S., Waterman, M.S.: Genomic mapping by fingerprinting random clones: a mathematical analysis. *Genomics* **2** (1988) 231–239
5. Bollobás, B.: Extremal graph theory. In Graham, R.L., Grötschel, M., Lovász, L., eds.: *Handbook of Combinatorics*. Volume II. Elsevier, Amsterdam (1995) 1231–1292
6. Reiman, I.: Über ein Problem von K. Zarankiewicz. *Acta Math. Sci. Hung.* **9** (1958) 269–279
7. Du, D.Z., Hwang, F.K.: *Combinatorial Group Testing and Its Applications*. 2nd edn. World Scientific, Singapore (2000)
8. Bruno, W.J., Knill, E., Balding, D.J., Bruce, D.C., Doggett, N.A., Sawhill, W.W., Stallings, R.L., Whittaker, C.C., Torney, D.C.: Efficient pooling designs for library screening. *Genomics* **26** (1995) 21–30
9. Beth, T., Jungnickel, D., Lenz, H.: *Design Theory*. 2nd edn. Cambridge University Press, UK (1999)
10. Barillot, E., Lacroix, B., Cohen, D.: Theoretical analysis of library screening using an  $n$ -dimensional strategy. *Nucleic Acids Res.* **19** (1991) 6241–6247
11. Kautz, W.H., Singleton, R.C.: Nonrandom binary superimposed codes. *IEEE Trans. Inform. Theory* **IT-10** (1964) 363–377
12. D’yachkov, A.G., Macula, Jr., A.J., Rykov, V.V.: New constructions of superimposed codes. *IEEE Trans. Inform. Theory* **IT-46** (2000) 284–290
13. Bouck, J., McLeod, M.P., Worley, K., Gibbs, R.A.: The Human Transcript Database: a catalogue of full length cDNA inserts. *Bioinformatics* **16** (2000) 176–177 (<http://www.hgsc.bcm.tmc.edu/HTDB/>).
14. Schuler, G.D.: Pieces of the puzzle: expressed sequence tags and the catalog of human genes. *J. Mol. Med.* **75** (1997) 694–698 (<http://www.ncbi.nlm.nih.gov/Unigene/>).
15. Jurka, J.: Repbase update: a database and an electronic journal of repetitive elements. *Trends Genet.* **16** (2000) 418–420 (<http://www.girinst.org/>).
16. Milosavljevic, A.: DNA sequence similarity recognition by hybridization to short oligomers (1999) U. S. patent 6,001,562.
17. Andersson, B., Lu, J., Shen, Y., Wentland, M.A., Gibbs, R.A.: Simultaneous shotgun sequencing of multiple cDNA clones. *DNA Seq.* **7** (1997) 63–70
18. Yu, W., Andersson, B., Worley, K.C., Muzny, D.M., Ding, Y., Liu, W., Ricafrente, J.Y., Wentland, M.A., Lennon, G., Gibbs, R.A.: Large-scale concatenation cDNA sequencing. *Genome Res.* **7** (1997) 353–358
19. Velculescu, V.E., Vogelstein, B., Kinzler, K.W.: Analysing uncharted transcriptomes with SAGE. *Trends Genet.* **16** (2000) 423–425

## Appendix

*Proof (Proposition 1).* The number of random shotgun reads from the row pool associated with the clone equals  $\frac{cmL}{2\ell}$ . By Equation (1), the probability that at least one of them aligns with the reference sequence equals

$$p_{\geq 1} = 1 - \left(1 - \frac{p_{\text{hit}}}{m}\right)^{\frac{cmL}{2\ell}} = 1 - \left(1 - \frac{M}{mL}\right)^{\frac{cmL}{2\ell}} \approx 1 - e^{-c\frac{M}{2\ell}}. \quad (4)$$

The probability that the row and column pools both generate reads aligning to the reference sequence equals  $p_{\geq 1}^2$ , as claimed.

*Proof (Proposition 2).* The number of hits for an index coming from a fixed row or column pool is distributed binomially with parameters  $n = \frac{cmL}{2\ell}$  and  $p = \frac{M}{mL}$ . Let  $\xi_r$  denote the number of hits coming from the clone's row pool, and let  $\xi_c$  denote the number of hits coming from the clone's column pool. Then

$$E_M = \mathbb{E}[\xi_r + \xi_c \mid \xi_r > 0, \xi_c > 0] = \mathbb{E}[\xi_r \mid \xi_r > 0] + \mathbb{E}[\xi_c \mid \xi_c > 0].$$

In order to calculate the conditional expectations on the right-hand side, notice that if  $\xi$  is a non-negative random variable, then  $\mathbb{E}\xi = \mathbb{E}[\xi \mid \xi > 0]\mathbb{P}\{\xi > 0\}$ . Since  $\mathbb{P}\{\xi_c > 0\} = \mathbb{P}\{\xi_r > 0\} = p_{\geq 1}$ ,

$$E_M = 2\mathbb{E}[\xi_r \mid \xi_r > 0] = \frac{2np}{p_{\geq 1}}$$

Resubstituting  $p$  and  $n$ , the proposition follows from Equation (4).

*Proof (Proposition 3).* Let  $n = \frac{cmL}{2\ell}$  be the number of reads from a pool and  $p = \frac{M}{\ell}$  the probability of a hit within a pool containing one of the clones. Then a false positive occurs if there are no hits in the row pool for one clone and the column pool for the other, and there is at least one hit in each of the other two pools containing the clones. The probability of that event equals

$$2\left((1-p)^n\right)^2\left(1 - (1-p)^n\right)^2.$$

Using the same approximation technique as in Proposition 1 leads to Equation (2).

*Proof (Theorem 1).* Let  $\mathcal{R}$  be an arbitrary rectangle in the array. We calculate the probability that  $\mathcal{R}$  is preserved after a random shuffling by counting the number of shufflings in which it is preserved. There are  $\binom{m}{2}$  choices for selecting two rows for the position of the preserved rectangle. Similarly, there are  $\binom{m}{2}$  ways to select two columns. There are 8 ways of placing the clones of the rectangle in the cells at the four intersections in such a way that the diagonals are preserved (see Figure 4). There are  $(m^2 - 4)!$  ways of placing the remaining clones on the array. Thus, the total number of shufflings in which  $\mathcal{R}$  is preserved equals

$$8\binom{m}{2}^2 (m^2 - 4)! = 8\left(\frac{m(m-1)}{2}\right)^2 (m^2 - 4)!.$$

Dividing this number by the number of possible shufflings gives the probability of preserving  $\mathcal{R}$ :

$$p = \frac{8\binom{m}{2}^2}{(m^2)(m^2 - 1)(m^2 - 2)(m^2 - 3)}. \quad (5)$$

For every rectangle  $\mathcal{R}$ , define the indicator variable  $I(\mathcal{R})$  for the event that it is preserved. Obviously,  $\mathbb{E}I(\mathcal{R}) = p$ . Using the linearity of expectations and Equation (5),

$$\mathbb{E}R(m) = \sum_{\mathcal{R}} \mathbb{E}I(\mathcal{R}) = \binom{m}{2}^2 p = \frac{1}{2} \cdot \frac{m^4(m-1)^4}{m^2(m^2-1)(m^2-2)(m^2-3)}. \quad (6)$$

Thus,

$$\mathbb{E}R(m) = \frac{1}{2} \left( 1 - \frac{4}{m} \left( 1 + o(1) \right) \right),$$

proving the theorem. Equation (6) also implies that  $\frac{\mathbb{E}R(m+1)}{\mathbb{E}R(m)} > 1$  for  $m > 2$ , and thus  $\mathbb{E}R(m)$  is increasing monotonically.

*Proof (Proposition 4).* Property 1 is trivial. For Property 2, notice that clone  $B_{a,b}$  is included in pool  $P_{i,a+ib}$  in every  $\mathcal{P}_i$ , and in no other pools. For Property 3, let  $P_{x_1,y_1}$  and  $P_{x_2,y_2}$  be two arbitrary pools. Each clone  $B_{a,b}$  is included in both pools if and only if

$$y_1 = a + bx_1 \quad \text{and} \quad y_2 = a + bx_2.$$

If  $x_1 \neq x_2$ , then there is exactly one solution for  $(a, b)$  that satisfies both equalities.

*Proof (Lemma 1).* Fix the first  $(k-1)$  coordinates of  $\mathbf{u}$  and let the last one vary from 0 to  $(q-1)$ . The  $i$ -th coordinate of the corresponding codeword  $\mathbf{c} = \mathbf{u}\mathbf{G}$  takes all values of  $\mathbb{F}_q$  if the entry  $\mathbf{G}[k, i]$  is not 0.

*Proof (Proposition 5).* The first part of the claim for the case  $w(\mathbf{x}) \geq k$  is a consequence of the error-correcting properties of the code. The second part of the claim follows from the MDS property.

*Proof (Proposition 6).* This proof generalizes that of Proposition 1. The number of random shotgun reads from one pool associated with the clone equals  $\frac{cmL}{n\ell}$ . By Equation (1), the probability that at least one of them aligns with the reference sequence equals

$$p_{\geq 1} = 1 - \left( 1 - \frac{p_{\text{hit}}}{m} \right)^{\frac{cmL}{n\ell}} \approx 1 - e^{-c\frac{M}{n\ell}}.$$

The probability that at least  $n_{\min}$  pools generate reads aligning to the reference sequence equals

$$\sum_{t=n_{\min}}^n \binom{n}{t} p_{\geq 1}^t (1 - p_{\geq 1})^{n-t}$$

proving the claim with  $p_0 = 1 - p_{\geq 1}$ .