

Pooled genomic indexing (PGI): analysis and design of experiments

Miklós Csűrös*[†]

Aleksandar Milosavljevic ^{†‡}

July 29, 2004

Abstract

Pooled Genomic Indexing (PGI) is a novel method for physical mapping of clones onto known sequences. PGI is carried out by pooling arrayed clones, and generating shotgun sequence reads from the pools. The shotgun sequences are compared to a reference sequence. In the simplest case clones are placed on an array and are pooled by rows and columns. If a shotgun sequence from a row pool and another shotgun sequence from a column pool match the reference sequence at a close distance, they are both assigned to the clone at the intersection of the two pools. Accordingly, 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 designs are described and analyzed, including transversal designs and designs from linear codes. 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.

*Département d'informatique et recherche opérationnelle, Université de Montréal, CP 6128 succ. Centre-Ville, Montréal, QC H3C 3J7, Canada. E-mail: csuros@iro.umontreal.ca. Phone: +1 (514) 343-6111 extension 1655, Fax: +1 (514) 343-5834.

[†]Bioinformatics Research Laboratory, Department of Molecular and Human Genetics, Baylor College of Medicine, Houston, TX 77030, USA.

[‡]Human Genome Sequencing Center, Department of Molecular and Human Genetics Baylor College of Medicine, Houston, TX 77030, USA. E-mail: amilosav@bcm.tmc.edu.

1 Introduction

We propose a novel method for physical mapping of clones onto known macromolecular sequences, called *Pooled Genomic Indexing* (PGI). Figure 1 illustrates a basic application of PGI to BAC¹ clone mapping. Starting with a BAC array, DNA is pooled together from clones in the same row (or column). A subclone library is prepared from every row and column pool. Random shotgun sequences (*reads*) are collected from each subclone library. The shotgun sequences are used to map individual clones to homologous sequences of a related organism. Specifically, shotgun sequences are compared to a database of reference sequences using standard alignment techniques: if a shotgun sequence from a row pool and a shotgun sequence from a column pool match the same reference sequence at a close distance, then both sequences are assigned (*deconvoluted*) to the clone at the intersection of the given row and column. Simultaneously, the clone is mapped onto the region of the reference sequence between the matches. Known reference sequences are thus turned into *indexes* to unsequenced homologous clones across species.

[Figure 1 about here.]

PGI indexes can guide the targeted sequencing of homologous regions, thereby making them available for comparative sequence analysis methods. Comparative sequence analysis has proven useful for the discovery of conserved regulatory regions, genes, and gene structure. PGI has two advantages over other targeted approaches to comparative sequencing (Thomas et al. 2002). First, PGI does not require diverse experimental procedures such as overgo probe design and hybridization. The shotgun sequencing of BAC pools provides all the necessary information for comparative mapping. Secondly, the depth of shotgun sequencing in PGI can be adjusted to a large range of evolutionary distances between organisms. In contrast, BAC-end sequencing for instance permits mapping across closely related organisms (Fujiyama et al. 2002).

Our reason for proposing PGI is motivated by recent advances in sequencing technology that allow shotgun sequencing of BAC pools. The Clone-Array Pooled Shotgun Sequencing (CAPSS) method (Cai et al. 2001) relies on clone-array pooling and shotgun sequencing of the pools. CAPSS assembles shotgun sequences from different pools into contigs. Contigs that contain shotgun sequences from a particular row pool and a particular column pool are deconvoluted to the clone at the intersection of the row and the column. In a clone-by-clone sequencing strategy (IHGSC 2001), the shotgun reads are collected for each clone separately. When pooling is used, the individual shotgun reads are not directly associated with the clones. The clone association information is recovered by comparing the shotgun sequences to each other (CAPSS) or to reference sequences (PGI). Despite the 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 the performance of CAPSS, indicating that improvements of one method are generally applicable to the other.

The purpose of this paper is three-fold. First, we introduce PGI, and argue that the pooling approach leads to a viable comparative mapping strategy. Secondly, we describe and analyze combinatorial methods for the pooling. Thirdly, we describe a probabilistic model in which different pooling schemes can be compared. Along with simulation results, the model can guide the selection of pooling designs and experimental parameters in mapping projects.

The paper is structured as follows. Section 2 describes a probabilistic model for the PGI method, and points out possible shortcomings due to redundancies among the clones, which cannot be resolved by the basic array-based pooling design. Section 3 discusses different experiment designs. In one line of extending the basic pooling design, §3.1 examines designs involving one array in which positions are left empty systematically in order to reduce problems caused by clone redundancies. In another way of extending the basic design, §3.2 and §3.3 propose the introduction of several arrays containing the same clone set in different layouts. Random layouts are discussed in §3.2, and deterministic layouts are analyzed in §3.3. A natural extension of the latter, designs based on error-correcting codes are examined in §3.4. Section 3 ends with a comparison of pooling designs and experiment parameters, based on our probabilistic analysis. Section 4 compares the pooling designs in simulated experiments involving 207 mouse and 625 rat clones,

¹Bacterial Artificial Chromosome

mapped to human sequences. Section 5 concludes the paper with a discussion of the results and an outlook to further research directions.

2 Probabilistic analysis

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 ℓ be the expected length of a read. Define the *read coverage* $c = \frac{F\ell}{NL}$. Assuming uniform coverage and a $m \times m$ array, the number of reads coming from a fixed pool equals $R = \frac{cmL}{2\ell}$.

2.1 Closely related species

PGI may be applied to map BACs of one species onto the genomic sequence of a closely related reference species. At low evolutionary distances, extremely low levels of pool shotgun coverage may suffice. At low levels of shotgun sequencing the question is whether or not a BAC is sampled at all. Moreover, in order for a clone to be indexed via PGI, informative reads originating from the clone must be sampled in multiple pools. We propose a simple model of clone indexing.

The probability that a particular BAC is sampled by at least one of the R reads in a pool equals

$$p_{\text{sample}} = 1 - \left(1 - \frac{1}{m}\right)^R \approx 1 - e^{-\frac{R}{m}} = 1 - e^{-c\frac{L}{2\ell}}. \quad (1)$$

The expected number of sampled BACs in a pool thus equals

$$D = mp_{\text{sample}} = m\left(1 - e^{-\frac{R}{m}}\right).$$

The fraction of sampled BACs (p_{sample}) exponentially approaches 1 as the ratio of reads per pool vs. pool complexity (R/m) increases. The following table gives a number of representative values.

R/m	0.75	1.0	1.25	1.5	1.75	2.0	3.0	4.0
p_{sample}	0.53	0.63	0.71	0.78	0.83	0.86	0.95	0.98

PGI deconvolution also requires that reads identify their homologous or orthologous (or both) positions across species. To estimate the probability of recognizing an orthologous position, we define the variable S as the fraction of randomly selected sequence fragments of a certain length that uniquely identify a position in the genome.

Similarly to Equation (1), we can obtain the probability that a particular BAC is sampled in a pool by an informative read:

$$p_{\text{hit}} = 1 - e^{-\frac{SR}{m}} = 1 - e^{-c\frac{SL}{2\ell}}. \quad (2)$$

A particular BAC can be mapped if it is sampled by informative reads in both pools it is included in. The following proposition follows from Equation (2).

Proposition 1. *The probability that a clone is mapped equals approximately*

$$p_S \approx \left(1 - e^{-\frac{SR}{m}}\right)^2 = \left(1 - e^{-c\frac{SL}{2\ell}}\right)^2.$$

Calculated from Proposition 1, the following table shows representative values for the fraction of mapped BACs as function of read coverage and similarity.

	$R/m = 0.75$	1	1.25	1.5	1.75	2	3	4	8
$S = 88\%$	$p_S = 0.233$	0.342	0.445	0.537	0.617	0.686	0.862	0.942	0.998
80%	0.204	0.303	0.400	0.488	0.568	0.637	0.827	0.920	0.997
70%	0.167	0.253	0.340	0.423	0.499	0.568	0.770	0.882	0.993
60%	0.131	0.204	0.278	0.352	0.423	0.488	0.697	0.827	0.984
50%	0.098	0.155	0.216	0.278	0.340	0.400	0.604	0.748	0.964

In order to establish some range of possible values for S as a function of evolutionary distance, the following simulated experiment was performed on the GoldenPath Human genome assembly (IHGSC 2001): random sequences of certain lengths (ranging from 100 to 200 to 500bp) were selected and randomly mutated at a specified mutation level (4, 6, 8, 10%); one thousand randomly selected mutated sequences were then used to search GoldenPath assembly (April 2001 version) using BLAT search (Kent 2002); the percentage of the cases S in which the top hit was the original position from which the sequence was selected is shown in the following table.

	read length $\ell=100$	200	500
percent difference = 4%	$S = 85\%$	87%	88%
6%	75%	79%	81%
8%	64%	70%	72%
10%	50%	58%	58%

In another experiment, we selected 521 chimpanzee (*Pan troglodytes*) BAC clones (all finished genomic sequences with length at least 50kbp in GenBank), simulated shotgun sequencing, and compared the shotgun sequences to the human genome sequence, in order to verify how correct our simple model of "informative reads" is. Shotgun sequences of expected length 100bp were generated as described in Section 4, mutating random substrings of the clones with a 1% substitution error. Each clone was sampled by 4 shotgun sequences on average. Shotgun sequences were compared to the human genome sequence (NCBI Build 34, after discarding alternative assembly of chromosome 7), using BLAST (Altschul et al. 1997) with the parameters of Section 4: minimum 40 bp match, minimum 60 score, maximum 10^{-5} E-value. Reads generating more than 12 matches were discarded, and so were matches involving the same human genome segment and more than 3 chimpanzee clones. (The latter criterion corresponds to resolvable ambiguities in two-array designs, see §3.3.) We found that 1543 (74%) of the reads were informative, in accordance with the table above given that the divergence between human and chimpanzee genomes is 5% (Britten 2002), and we corrupted the shotgun sequences with an additional 1% error. We further analyzed the matches to see if this ratio of "informative reads" is uniform across clones. While it is obviously an approximation, we argue that it is a fairly accurate one: we use a quartile-based statistical test for this purpose. For $\alpha = 25\%, 50\%, 75\%$, we calculated the number of clones that had at least α fraction of their reads mapped to the genome. If every read can be mapped to the human genome with a uniform $p = 1543/2084 \approx 0.74$ probability, then the expected number of clones with at least α fraction of mapped reads is

$$N_\alpha = N \sum_{k=0}^{\infty} \binom{N}{k} q^k (1-q)^{N-k} \sum_{j=\lceil \alpha k \rceil}^k \binom{k}{j} p^j (1-p)^{k-j},$$

where $N = 521$ is the number of clones, and $q = 4/N$, the probability that a given read originates from a given clone. Hence $N_{0.25} = 493$, $N_{0.5} = 459$, and $N_{0.75} = 292$. These expected values were compared to the actual number \tilde{N}_α of clones that have $\geq \alpha$ fraction of mapped reads. In the experiments we measured $\tilde{N}_{0.25} = 482$, $\tilde{N}_{0.5} = 441$, and $\tilde{N}_{0.75} = 289$, which are fairly close to the predicted values.

2.2 Distantly related species

When mapping, for instance, one mammalian genome onto another, PGI relies on finding similarities between conserved regions. In such an application, a significant portion of the shotgun reads, namely those that do not match any reference sequence, will not be useful for the purposes of comparative mapping. Furthermore, since reads are randomly distributed along the clones, it is not certain that homologies between reference sequences and clones are detected. However, this probability increases rapidly with the number of shotgun sequences. Consider the particular case of detecting homology between a given reference sequence and a clone. A shotgun sequence of length λ 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 shotgun sequence of length λ can begin and produce a significant

alignment. The probability of a hit for a fixed length λ 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 shotgun sequence matching the reference sequence equals

$$p_{\text{hit}} = \mathbb{E} \frac{M(\lambda)}{L - \lambda + 1} \approx \frac{M}{L}, \quad (3)$$

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 λ , 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 $M = g + e(\ell - 2o)$.

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 $\binom{cmL}{2\ell} \left(\frac{p_{\text{hit}}}{m}\right) = \frac{cM}{2\ell}$. Propositions 2, 3, and 4 rely on the properties of this distribution, using Poisson approximation techniques.

Proposition 2. *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. The number of random shotgun reads from the row pool associated with the clone equals $\frac{cmL}{2\ell}$. By Equation (3), 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. \square

By Proposition 2, the probability of false negatives decreases exponentially with the read coverage, as $1 - p_M \rightarrow e^{-c\frac{M}{\ell}}$. The number of hits for a detected index can be calculated similarly.

Proposition 3. *The expected number of hits that comprise a detected index with effective length M equals approximately*

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

Proof. 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 ξ_r denote the number of hits coming from the clone's row pool, and let ξ_c denote the number of hits coming from the clone's column pool. By the independence of the pools,

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

In order to calculate the conditional expectations on the right-hand side, notice that if ξ 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). \square

Notice that Proposition 3 provides a means to estimate the unknown effective length of a detected index.

In what follows we concentrate on the case of large distances. Nevertheless, the results apply immediately to the case of short distances if we recognize that the effective length of an index plays the same rôle in the former as the percentage of informative reads in the latter. Consequently, by substituting (SL) for M in the formulas we obtain the corresponding results for short distances. In addition to mapping genomic clones to a reference genome sequence, the focus on effective length captures more general cases such as the mapping of genomic clones to transcribed reference sequences, and the mapping of cDNA clones.

2.3 Ambiguous indexes and false positives

The success of indexing in the PGI method depends on the possibility of deconvoluting the local alignments. In the simplest case, a homology between a clone and a reference sequence is recognized by finding alignments with shotgun sequences 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). Typical cases include when the clones overlap, contain similar genes, or contain similar repeat sequences. Subsequently, close alignments may be found between a reference sequence and shotgun sequences 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 2a.

[Figure 2 about here.]

In addition to the problem of ambiguous deconvolution, multiple homologies may also lead to incorrect indexing at low coverage levels. Referring to the example of Figure 2b, assume that the clones B_{12} and B_{21} contain a particular homology. If the read coverage is low, it may happen that the only homologies found are from row R_1 and column C_1 . In that case, the clone B_{11} 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 4. *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*

$$p_{\text{FP}} \approx 2e^{-c\frac{M}{\ell}} \left(1 - e^{-c\frac{M}{2\ell}}\right)^2. \quad (5)$$

Proof. 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 2 leads to Equation (5). □

3 Pooling designs

Clearly, in order to reduce the number of ambiguous indexes and false positives, we need to depart from the idea of using a simple single-array design. In a general setting, 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 pooling design is described by the *pooling matrix* \mathbf{M} , which is an $N \times K$ matrix that represents the incidence structure of the pooling. The entry $\mathbf{M}[i, j]$ equals one if clone B_i is included in P_j , otherwise it is 0. The development of incidence structures with various properties is an active research area of combinatorics (Beth et al. 1999), and in particular, combinatorial group testing (Du and Hwang 2000). Clone pooling designs have been successfully used for hybridization-based physical mapping (Barillot et al. 1991; Bruno et al. 1995; Balding et al. 1996). However, PGI raises different issues on evaluating pooling designs from those encountered in hybridization. In particular, the probabilities of false positives and negatives (erroneous and failed indexing of a clone, respectively) depend on the design, while analysis of pooling for the purposes of hybridization generally assumes fixed probabilities (Bruno et al. 1995).

Laboratory technology involved in pooling and cloning have implications on the practicality of different pooling designs. First, it is useful if the design can be implemented in a simple way using clone arrays, by pooling rows and columns together as it reduces errors in protocols carried out either by humans or by robots. Secondly, it is advantageous to strive to minimize pool sizes, and the number of pools a clone is represented in, in order to obtain uniform clone representations by the shotgun sampling. With the exception of designs introduced in §3.4, all described pooling schemes are implemented by row and column pooling of arrayed clones.

The simplest conceptual improvement to the basic array-based pooling design is to distribute the clones on more than one array, which reduces ambiguities resulting from overlapping clones on the same array. Sparse arrays, presented in §3.1, provide a pooling design with a single array, in which specific cells are left empty. Another possibility is to repeat the pooling using different rearrangements on arrays of the same size. Random shuffling, presented in §3.2, can be used with arrays of any geometry. Transversal designs, presented in §3.3, provide an optimal way of rearranging the clones on square arrays. Generalizing transversal designs, §3.4 discusses designs based on linear error-correcting codes. Finally, §3.5 compares the different pooling designs in light of their characteristics for indexing success and ambiguity resolution.

3.1 Sparse arrays

In order to examine how to avoid ambiguities when using only one array, we introduce a graph representation. 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. 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.

[Figure 3 about here.]

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 two sparse arrays. 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 (Bollobás 1995). It is known (Lovász 1993) that if $N > \frac{K}{4}(1 + \sqrt{4K + 3})$, then \mathcal{G}^* does contain a cycle of length four, hence $c_K K^{3/2} \geq N$ where $\frac{1}{2} \leq c_K \leq c_3 = 3^{-\frac{1}{2}}$ and $c_K \rightarrow 1/2$.

To design optimal sparse arrays, we use an idea of Reiman (1958) that relies on finite combinatorial geometries. Using a projective or affine plane, rows are associated with points, and columns are associated with lines. A clone can be placed in a cell if the corresponding row's point is incident to the corresponding

column's line. By the properties of the combinatorial geometries, each pair of points is incident to one line, and every pair of lines intersect at at most one point, producing a sparse array.

For the sake of completeness, we describe in detail how an affine plane defines the pooling. Let m is 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. The number m is the size of a pool, i.e., the 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 . Figure 3b shows an example with $m = 7$.

3.2 Random shuffling

The principal advantage of PGI is the reduced cost of library preparations due to pooling. The number of pools can be further reduced by departing from the two-dimensional structure of arrayed pooling, as proposed by Barillot et al. (1991). A three-dimensional pooling strategy, for instance, can be based on a set of $m \times m$ arrays, arranged in m layers. In this case, in addition to row and column pools, layer pools are also defined, so that each clone is included in three pools. Specifically, the clone in row i and column j in layer l is included in row pool i , column pool j , and layer pool l . Such a cube-like design combines m^3 clones in $3m$ pools, an improvement over simple arrayed pooling in terms of number of libraries. However, this reduction comes at the price of decreased indexing success: in order to deconvolute an index to a clone, one needs hits from all the three pools. Applying the technique of Proposition 2, the probability of that event can be approximated as $p_M \approx \left(1 - e^{-c \frac{M}{3\ell}}\right)^3$. Figure 5 shows that in order to achieve the same indexing success as with arrayed pooling, the read coverage c must be increased by a factor of two to three. Moreover, three-dimensional pooling aggravates the problems of false positives and ambiguities. Finally, the number of pools is not reduced as dramatically as it would seem at first sight: using 48×48 arrays in 48 layers, about 110,000 clones are pooled in 144 pools. The same number of clones can be laid out on a 333×333 array, resulting in 666 pools.

A useful idea is to repeat the pooling with the clones reshuffled on two or more arrays of the same size. Taking the example of Figure 2a, 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_{12} is in cell (R'_1, C'_2) and B_{21} is in cell (R'_2, C'_1) , and alignments with fragments from the pools for R_i , C_i , R'_i , and C'_i are found, except if the clones B_{11} and B_{22} got assigned to cells (R'_1, C'_1) and (R'_2, C'_2) . Figure 4 shows the possible placements of clones in which the ambiguity is not resolved despite the shuffling. Our aim is to find shufflings in which ambiguities depicted in Figure 4 do not occur. We prove that ambiguities caused by two clones can be resolved with high probability when using random layouts.

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).

Proposition 5. *Let $R(m)$ be the number of preserved rectangles on a $m \times m$ array after a random shuffling. Then, for the expected value $\mathbb{E}R(m)$,*

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

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

[Figure 4 about here.]

Proof. 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 eight 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} :

$$p = \frac{8 \binom{m}{2}^2}{(m^2)(m^2 - 1)(m^2 - 2)(m^2 - 3)}. \quad (6)$$

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 (6),

$$\mathbb{E}R(m) = \sum_{\mathcal{R}} \mathbb{E}I(\mathcal{R}) = \binom{m}{2}^2 p = \frac{1}{2} \cdot \frac{m^4(m-1)^4}{m^2(m^2-1)(m^2-2)(m^2-3)}. \quad (7)$$

Thus,

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

proving the claim. Equation (7) 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. \square

Consequently, the expected number of preserved rectangles equals $\frac{1}{2}$ asymptotically. Proposition 5 also implies that with significant probability, a random shuffling produces an array with at most one preserved 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.

Practical considerations may constrain pooling designs to those that can be implemented using arrays of certain sizes such as the standard 8×12 geometry or its multiples. Proposition 5 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 in conjunction with non-square arrays.

3.3 Transversal designs

Random shufflings can be used to avoid ambiguities caused by two overlapping clones, as shown in 3.2. For higher order ambiguities, we consider *collinear* clones. A clone pair is called collinear if they are placed in the same row or column on one of the arrays. A pooling design based on shuffled arrays satisfies the *unique collinearity* condition, if every pair of clones appears in the same row or same column at most once.

A simple argument (e.g., Du and Hwang 2000) shows that if one uses d shuffled arrays with the unique collinearity condition, every set of up to $(2d - 1)$ identical or overlapping clones is uniquely identified by the pools they are included in, and thus indexes to $(2d - 1)$ identical clones can be deconvoluted. Let \mathcal{B} be a set with at most $(2d - 1)$ identical or overlapping clones, and let B' be an arbitrary clone that is not in \mathcal{B} . The clone B' is collinear at most once with every element of \mathcal{B} , and thus at least one of the $2d$ pools B' is included in contains no clone from \mathcal{B} . Hence, B' must be excluded from deconvolution of an index to \mathcal{B} . Similarly, for an arbitrary subset $\mathcal{B}' \subset \mathcal{B}$, a clone $B \in \mathcal{B} - \mathcal{B}'$ appears in at least one pool that contains no element of \mathcal{B}' , and thus B must be included in the deconvolution. Consequently, every clone of \mathcal{B} is uniquely indexed when using d arrays.

Shuffled arrays with the unique collinearity condition were proposed by Barillot et al. (1991) for hybridization-based clone mapping. It is known (Du and Hwang 2000) that the existence of $d > 1$ arrays of size $m \times m$ is equivalent to the existence of $2(d - 1)$ of mutually orthogonal Latin squares (Beth et al. 1999) of order m . Accordingly, an affine plane of order q can be used to produce up to $q/2$ arrays of size $q \times q$ satisfying unique collinearity (Raghavarao 1971). For completeness' sake, we describe here such a design in detail. Let q be a prime power. Based on results of design theory (Beth et al. 1999), 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 associated with elements of \mathbb{F}_q^2 , numbered as $P_{x,y}$: $x, y \in \mathbb{F}_q$. Define the pool set $\mathcal{P}_i = \{P_{i,y} : y \in \mathbb{F}_q\}$ for all i . Clones are also associated with elements of \mathbb{F}_q^2 , numbered 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 6. *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.

(Proven by Raghavarao (1971) among others.)

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. Based on 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.

A *transversal design* is a pooling design for which pools are partitioned into pool sets satisfying Properties 1–3 of Proposition 6. Transversal designs with an even number of pool sets correspond to shuffled arrays with unique collinearity and vice versa. For a pooling design with n pool sets, every clone is included in n pools, and any subset with at least 2 of those pools uniquely identifies the clone. The success of indexing in the general case is shown by the following claim.

Proposition 7. *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 $k \leq n$ of these pools uniquely determines the clone, then the probability that the index is detected equals*

$$p_M = \sum_{t=k}^n \binom{n}{t} (1 - p_0)^t p_0^{n-t} \quad (8)$$

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

Proof. This proof generalizes that of Proposition 2. The number of random shotgun reads from one pool associated with the clone equals $\frac{cmL}{n\ell}$. By Equation (3), the probability that at least one of them aligns with the reference sequence equals

$$p_{\geq 1} = 1 - \left(1 - \frac{p_{\text{hit}}}{m}\right)^{\frac{cmL}{n\ell}} \approx 1 - e^{-c \frac{M}{n\ell}}. \quad (9)$$

The probability that at least k pools generate reads aligning to the reference sequence equals

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

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

3.4 Designs from linear error-correcting codes

Here we discuss pooling designs based on linear error-correcting codes, relying on results of Kautz and Singleton (1964). Our goal is to define pooling designs in which each clone is included in n pools, with the property that any k -subset of those pools uniquely identifies the clone. In other words, we aim to generalize the “2-out-of- n ” property of transversal designs to “ k -out-of- n .”

An *error-correcting code* (MacWilliams and Sloane 1977) over a finite field \mathbb{F}_q is a set \mathcal{C} of vectors over \mathbb{F}_q with length n ; i.e., $\mathcal{C} \subseteq \mathbb{F}_q^n$. The elements of \mathcal{C} are called *codewords*, and the length n of the codewords is called the *code length*. The code \mathcal{C} has *minimum distance* d if two codewords always differ in at least d coordinates.

An arbitrary code \mathcal{C} over a finite field \mathbb{F}_q with length n can be employed to define a pooling design in the following manner. Pools are associated with elements of $\{1, \dots, n\} \times \mathbb{F}_q$, denoted by $P_{i,x} : i \in \{1, \dots, n\}, x \in \mathbb{F}_q$. Clones are associated with the codewords. The clone corresponding to the codeword \mathbf{c} is included in pool $P_{i,x}$ if the i -th coordinate of \mathbf{c} equals x . Obviously, each clone is included in exactly n pools. If \mathcal{C} has minimum distance d , then any subset with at least $k = n - d + 1$ of those n pools uniquely identifies the clone. This method defines a pooling with $K = qn$ pools for $N = |\mathcal{C}|$ clones. Looking at the rows of the corresponding pooling matrix \mathbf{M} , this pooling method relies on a mapping $f: \mathbb{F}_q^n \mapsto \mathbb{F}_2^{qn}$ of q -ary vectors onto binary vectors, by which the row vectors are obtained from the codewords.

A possible shortcoming of this mapping is that it may happen that the pools have unequal sizes, or, in other words, that the columns of \mathbf{M} have different binary weights. However, for a class of codes, called linear codes, the pool sizes are balanced. 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 row vector \mathbf{u} of length k over \mathbb{F}_q , a codeword \mathbf{c} is generated by calculating $\mathbf{c} = \mathbf{u}\mathbf{G}$. The code 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 claim.

Proposition 8. *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 pooling matrix \mathbf{M} be defined by mapping the first N codewords $\{\mathbf{c}^{(i)} = \mathbf{u}^{(i)}\mathbf{G} : i = 1, \dots, N\}$ onto binary vectors via f . 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. Let $\mathbf{u}^{(t)}, \mathbf{u}^{(t+1)}, \dots, \mathbf{u}^{(t+q-1)}$ be length k vectors that are consecutive in the lexicographic enumeration, and differ only in their last coordinates. Consider the corresponding clones associated with the codewords $\mathbf{c}^{(t)} = \mathbf{u}^{(t)}\mathbf{G}, \dots, \mathbf{c}^{(t+q-1)} = \mathbf{u}^{(t+q-1)}\mathbf{G}$. Denote the elements of the code’s generator matrix \mathbf{G} by $g_{i,j}$ with row indices $i = 1, \dots, k$ and column indices $j = 1, \dots, n$. The clone associated with an arbitrary codeword $\mathbf{c}^{(t')}$ is included in pool $P_{j,x}$ if and only if the j -th coordinate of $\mathbf{c}^{(t')}$ equals x , i.e., if

$$\sum_{i=1}^k \mathbf{u}_i^{(t')} g_{i,j} = x.$$

Let $y_j = \sum_{i=1}^{k-1} \mathbf{u}_i^{(t')} g_{i,j}$. By our selection of t , y_j has the same value for $t' = t, \dots, t + q - 1$. Thus the clone associated with $\mathbf{c}^{(t')}$ is included in every pool $P_{j,x_{t',j}}$ with $x_{t',j} = \mathbf{u}_k^{(t')} g_{k,j} + y_j$. If $g_{k,j}$ is not 0, then the set

$\{x_{t,j}, \dots, x_{t+q-1,j}\}$ contains all elements of \mathbb{F}_q for every j . Consequently, every pool contains exactly one of the clones associated with the codewords $\mathbf{c}^{(t)}, \dots, \mathbf{c}^{(t+q-1)}$. Let t_0 be the largest integer such that $t_0 \leq N$ and it is divisible by q without remainder. If the lexicographic enumeration of all vectors is truncated at the N -th element, the first $t_0 - 1$ clones contribute equally to every pool, and the remaining $(N - t_0 + 1)$ clones are included at most once in every pool. Hence every pool contains (t_0/q) or $(1 + t_0/q)$ clones. \square

We have no knowledge of a similar statement to Proposition 8 in the literature. It is useful in our context since it amounts to a simple way of selecting the N codewords for the N clones.

An important class of linear codes is the family of Reed-Solomon codes, which are particularly interesting for our purposes. 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 $\alpha_1, \alpha_2, \dots, \alpha_n$ be different non-zero elements of \mathbb{F}_q . The generator matrix \mathbf{G} of the RS(n, k) code is

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ \alpha_1 & \alpha_2 & \dots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \dots & \alpha_n^2 \\ \dots & \dots & \dots & \dots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \dots & \alpha_n^{k-1} \end{pmatrix}.$$

By using a mapping $f: \mathbb{F}_q^n \mapsto \mathbb{F}_2^{qn}$ for the first $N \leq q^k$ codewords, as before, a pooling design is obtained with N clones and $K = qn$ pools. This design has many advantageous properties. Since $\alpha_i^{k-1} \neq 0$ in the last row of \mathbf{G} , Proposition 8 applies and thus each pool has about the same size $m = \frac{N}{q}$. Each clone is included in n pools, and by the MDS property, a k -subset of those pools uniquely identifies the clone. Moreover, it can be shown (Kautz and Singleton 1964) that if $t = \lfloor \frac{n-1}{k-1} \rfloor$, then any t -set of clones are uniquely identified by the pools they are included in, i.e., there are no ambiguities in deconvoluting indexes comprising hits that originate from up to t clones.

For a pooling design based on the RS(n, k) code, define the pool sets $\mathcal{P}_i = \{P_{i,x} : x \in \mathbb{F}_q\}$ for all $i = 1, \dots, n$. Such pooling designs generalize transversal designs as shown by the following result.

Proposition 9. *The pooling design based on the RS(n, k) code as 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 k -set of pools that belong to different pool sets, there is exactly one clone that is included in all pools.*

Proof. Property 1 is trivial. For Property 2, notice that the clone associated with the codeword \mathbf{c} is included in the pool P_{i,\mathbf{c}_i} in every \mathcal{P}_i , and in no other pools. Property 3 follows from the MDS property of the Reed-Solomon code: there is exactly one codeword that matches at least k coordinates of an arbitrary length n vector. \square

Remark. The design based on the RS($n, 2$) code gives the same transversal design we described in §3.3. When $k = 2$, \mathbf{G} has the structure

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ \alpha_1 & \alpha_2 & \dots & \alpha_n \end{pmatrix}.$$

Therefore, the pool set \mathcal{P}_i of the RS($n, 2$)-based design corresponds to the pool set \mathcal{P}_{α_i} of the transversal design, and the clone for the codeword $\mathbf{c} = [a, b]\mathbf{G}$ of the code-based design corresponds to clone $B_{a,b}$ of the transversal design.

3.5 Comparison of pooling designs

Various pooling designs (Barillot et al. 1991; Bruno et al. 1995; Chateaufneuf et al. 1999; Macula 1999; Ngo and Du 2002) have been proposed in recent years, mostly in the context of BAC library screening through hybridization. Many group testing techniques (Raghavarao 1971; Du and Hwang 2000) are also applicable to the construction of pooling designs. The main advantage of pooling designs is the reduced cost of experiments, but when comparing pooling designs, other features may also need to be considered. Two features generally sought for in pooling designs are error-correcting and ambiguity-resolving capabilities.

[Figure 5 about here.]

The power of resolving ambiguities is usually termed as the t -cover-free property: for all t -sets of clones, the set of pools in which they are included is a set that is not a superset of the set of pools in which an arbitrary separate clone is included. An example is the pooling design based on a Reed-Solomon code: the $RS(n, k)$ code leads to a t -cover-free design with $t = \lfloor \frac{n-1}{k-1} \rfloor$ (Kautz and Singleton 1964). Another example is a transversal design with $(t+1)$ pool sets: such a design is t -cover-free. Asymptotic properties of t -cover-free designs have been extensively studied (Kautz and Singleton 1964; D'yachkov and Rykov 1982; Erdős et al. 1985; Hwang and Sós 1987; Ruzinkó 1994; Csűrös and Ruzinkó 2004).

We characterized the error-correcting property of designs as the k -out-of- n property: every clone is included in n pools and k out of those pools uniquely identify the clone. In pooled hybridization experiments both false positive and false negative errors are to be considered. However, the probability of false positives in pooled shotgun experiments, i.e., that a significant local alignment is produced solely by read errors, is negligible. Another difference from most group testing applications is that the test error depends on the pooling design as described in Proposition 7. Figure 5 compares the indexing failure probabilities for several pooling designs with the k -out-of- n property. An important observation is that for a fixed k and coverage level, even as n increases, the indexing failure probability stays positive, determined by the value of k . This positive limit is the value of the Poisson cumulative distribution function with expected value cM/ℓ at $(k-1)$, as shown by the following result.

Proposition 10. *Consider an index with effective length M between a clone and a reference sequence. For a family of pooling schemes such that the clone appears in n pools, and a set of at least $k \leq n$ of these pools uniquely determines the clone, where k is kept constant while $n \rightarrow \infty$,*

$$\lim_{n \rightarrow \infty} p_M = 1 - e^{-c\frac{M}{\ell}} \sum_{t=0}^{k-1} \frac{\left(c\frac{M}{\ell}\right)^t}{t!}. \tag{10}$$

Proof. The key observation in the proof is that in Equation (9),

$$\lim_{n \rightarrow \infty} np_{\geq 1} = c\frac{M}{\ell}.$$

Then, by standard results in probability theory (Rényi 1970), the binomial distribution with parameters n and $p_{\geq 1}$ converges to the Poisson distribution with expected value $c\frac{M}{\ell}$. □

[Table 1 about here.]

[Table 2 about here.]

Propositions 7 and 10 suggest that “advanced” pooling designs with large k and n are not necessarily good since the higher indexing failure probability can be reduced only by increasing the read coverage: what we save on library preparations, we may lose on shotgun read collection. Tables 1 and 2 compare designs equivalent to the use of 24×24 and 96×96 arrays not only in their indexing success probabilities but also in terms of their cost benefits. While three-dimensional pooling needs the fewest number of pools, it is impractical as its indexing failure is much more likely than other design’s, it has no ambiguity-resolving

capabilities, and its pool sizes grow rapidly with the number of clones — it is unclear at this point how to achieve uniform clone representation in pools of several hundred clones. The designs based on Reed-Solomon codes seem also impractical due to large pool sizes, and to the large number of pools a clone needs to be included in. Nevertheless, their indexing failure probabilities are comparable to those of arrayed pooling designs, and they provide the best ambiguity-resolving capabilities. Sparse array designs need many pools for realistic sized projects and thus have limited benefits, especially in light of their modest ambiguity-resolving features. In particular, transversal designs with three arrays outperform the sparse designs in every aspect. On the other hand, three arrays may be prohibitively expensive for small (24×24) arrays. In conclusion, a transversal design with two arrays has the most advantages. In case of mammalian genomes, the 9216 clones laid out on a 96×96 array represent 0.5–1X clone coverage of the genome, and thus are unlikely to contain many ambiguities with four or more overlapping clones. Consequently, comparative mapping of mammalian genomes can be performed using groups of 9216 clones, where every group is pooled with a two-array transversal design.

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 (IHGSC 2001), the Human Transcript Database (HTDB) (Bouck et al. 2000), and the Unigene database of human transcripts (Schuler 1997). Local alignments were computed using BLASTN (Altschul et al. 1997) 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 least 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. As a result, 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 (Jurka 2000), 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.

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 14×15 arrays for (random) double shuffling, a 39×39 sparse array, shown in Figure 3a, two 16×16 arrays for transversal design, and a design based on the RS(6,3) code with 42 pools.

[Table 3 about here.]

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 3. 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 advanced design 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.

[Figure 6 about here.]

In order to explore the effect of read 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.

[Table 4 about here.]

Table 4 aims to assess how much information is lost solely by pooling. Using the same shotgun sequences generated in the simulation experiments, we performed the clone indexing with the knowledge of which clone the shotgun sequences originate from. BLAST parameters were the same as the ones used in the pooled experiments, and we discarded reads that matched more than 12 Unigene sequences. We also discarded indexes to more than three clones to exclude the effects of ambiguities. The table indicates that only 43–58% of the possible indexes are missed due to the pooling.

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 5.

[Table 5 about here.]

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.

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 subclone 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 it relies on shotgun libraries and sequences that can be reused in the sequencing phase.

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. Since our purpose of performing the experiments was to show a proof of the PGI concept, we have not repeated them with different sets of shotgun sequences. Moreover, we argue that different clones are sampled fairly independently: dependencies are restricted to clones that share the same pool. Consequently, the numbers of Table 3 are sums of many indicator variables with limited dependence, i.e., their variance should be small. This argument is further supported by Figure 6, which plots ten independent samplings for each coverage level, and by Table 4, which shows the results of five independent samplings. In both cases, the numbers vary very little across experiments. The experiments support our theoretical analysis in §3.5: among the studied pooling designs, two-array transversal designs balance cost and performance the best.

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. Due to the low level of chromosomal rearrangements across mammals, the order of BACs mapped to a reference mammalian genome corresponds almost exactly to the order of BACs along the indexed mammalian genome. The BAC ordering can be used in the complete sequencing of the indexed organism. Moreover, the reference sequences may guide the sequence assembly in a comparative sequencing approach (Milosavljevic 1999).

PGI constructs comparative BAC-based physical maps at a fraction (on the order of 1% for closely related species) of the cost of full genome sequencing. PGI requires only minor changes in the BAC-based sequencing pipeline already established in sequencing laboratories. BAC pooling is the key to the economy of PGI. The depth of shotgun sequencing 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 of the pooling scheme. In other words, pooling by itself does not represent a significant overhead, and yet leads to a comprehensive and accurate comparative physical map.

PGI is not limited to the mapping of BAC clones. Other applications currently include the mapping of arrayed cDNA clones onto genomic, or full or partial cDNA sequences within and across species, and the mapping of bacterial genomic clones across different bacterial strains.

BACs can be comparatively mapped by BAC-end sequencing (Venter et al. 1996) and subsequent anchoring of BAC ends onto orthologous regions of a related genome (e.g., Fujiyama et al. 2002). Due to the size of clone insert, a BAC end sequence read is significantly more costly than a read from a short-insert shotgun library that was prepared from a BAC pool. On the other hand, BAC-end sequencing does not incur the library preparation cost for each BAC pool. In contrast to BAC-end sequencing, PGI maps shotgun sequences obtained by randomly sampling the whole BAC, not just the ends. The sequencing depth may be adjusted to fit evolutionary distance. Genomic rearrangements may be robustly detected via PGI by mapping two halves of a single BAC spanning a breakpoint to two distant locations in the genome of a related organism. In contrast, confident detection of a breakpoint by BAC-end sequencing involves mapping of at least two BACs across a breakpoint.

Two new experimental aspects of PGI technology are currently in development. First, sampling efficiency of PGI is increased by an order of magnitude by sequencing short mappable sequence tags (50–150 bp). This is accomplished by breaking the pooled DNA into short fragments (“tags”), forming concatemers by ligation, and by sequencing the concatemers (Andersson et al. 1997; Yu et al. 1997; Velculescu et al. 2000). Assuming

a sequence read of 750 bp and tag size of 50–150 bp, a total of 5 to 15 different clones are sampled in a single sequencing reaction. This method, which we refer to as Short-Tag PGI is particularly useful when the clone sequences and reference sequences are highly similar, as is the case when mapping across primates. The second improvement involves the pooling of the pools prior to shotgun library preparation. Shotgun reads (concatemers of tags from an individual pool) are assigned back to the original pool through the sequencing of a short pool-specific “barcode”, which is introduced into the concatemers prior to pooling. This second improvement has the potential to reduce the library preparation cost by a factor of 24 to 192. Taken together, the two improvements have the potential to produce by far the most economical method for comparative physical mapping of BAC clones.

Acknowledgments

We are grateful to Richard Gibbs and George Weinstock for sharing pre-publication information on CAPSS and for useful comments, to Andrew Jackson and Dragomir Popovic for their help with the simulation experiments, and to Paul Havlak and David Wheeler for contributing database access and computational resources at HGSC. We also thank an unnamed referee whose careful reading and comments helped us improve the paper. This work was supported by grants RO1 HG02583-01 and U01 RR18464 from the National Institutes of Health, 250391-02 from the Natural Sciences and Engineering Research Council of Canada, Howard Hughes Medical Institute faculty startup funds, and Université de Montréal faculty startup funds.

References

- Altschul, S. F., T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* 25, 3389–3402.
- Andersson, B., J. Lu, Y. Shen, M. A. Wentland, and R. A. Gibbs (1997). Simultaneous shotgun sequencing of multiple cDNA clones. *DNA Seq.* 7, 63–70.
- Balding, D. J., W. J. Bruno, E. Knill, and D. C. Torney (1996). A comparative survey of non-adaptive pooling designs. In T. Speed and M. S. Waterman (Eds.), *Genetic Mapping and DNA Sequencing*, Volume 81 of *IMA volumes in mathematics and its applications*, pp. 133–154. New York: Springer.
- Barillot, E., B. Lacroix, and D. Cohen (1991). Theoretical analysis of library screening using an n -dimensional strategy. *Nucleic Acids Res.* 19, 6241–6247.
- Beth, T., D. Jungnickel, and H. Lenz (1999). *Design Theory* (2nd ed.). UK: Cambridge University Press.
- Bollobás, B. (1995). Extremal graph theory. In R. L. Graham, M. Grötschel, and L. Lovász (Eds.), *Handbook of Combinatorics*, Volume II, Chapter 23, pp. 1231–1292. Amsterdam: Elsevier.
- Bouck, J., M. P. McLeod, K. Worley, and R. A. Gibbs (2000). The Human Transcript Database: a catalogue of full length cDNA inserts. *Bioinformatics* 16, 176–177.
- Britten, R. (2002). Divergence between samples of chimpanzee and human DNA sequences is 5%, counting indels. *Proc. Natl. Acad. Sci. USA* 99, 13633–13635.
- Bruno, W. J., E. Knill, D. J. Balding, D. C. Bruce, N. A. Doggett, W. W. Sawhill, R. L. Stallings, C. C. Whittaker, and D. C. Torney (1995). Efficient pooling designs for library screening. *Genomics* 26, 21–30.
- Cai, W.-W., R. Chen, R. A. Gibbs, and A. Bradley (2001). A clone-array pooled strategy for sequencing large genomes. *Genome Res.* 11, 1619–1623.
- Chateaneuf, M. A., C. J. Colbourn, D. L. Kreher, E. R. Lamken, and D. C. Torney (1999). Pooling, lattice square, and Union Jack designs. *Ann. Combin.* 3, 27–35.
- Csűrös, M. and M. Ruzsínkó (2004). Single-user tracing superimposed codes. In *Proceedings of 2004 IEEE International Symposium on Information Theory*. To appear.
- Du, D.-Z. and F. K. Hwang (2000). *Combinatorial Group Testing and Its Applications* (2nd ed.). Singapore: World Scientific.
- D'yachkov, A. G. and V. V. Rykov (1982). Bounds on the length of disjunctive codes. *Problemi Peredachi Informatsii* 18(3), 7–13.
- Erdős, P., P. Frankl, and Z. Füredi (1985). Families of finite sets in which no set is covered by the union of r others. *Israel J. Math.* 51, 79–89.
- Fujiyama, A., H. Watanabe, A. Toyoda, T. D. Taylor, T. Itoh, S.-F. Tsai, H.-S. Park, M.-L. Yaspo, H. Lehrach, Z. Chen, G. Fu, N. Saitou, K. Osoegawa, P. J. de Jong, Y. Suto, M. Hattori, and Y. Sakaki (2002). Construction and analysis of a human-chimpanzee comparative clone map. *Science* 295(5552), 131–134.
- Hwang, F. K. and V. T. Sós (1987). Non-adaptive hypergeometric group testing. *Stud. Sci. Math. Hungar.* 22, 257–263.
- IHGSC (International Human Genome Sequencing Consortium) (2001). Initial sequencing and analysis of the human genome. *Nature* 609(6822), 860–921.
- Jurka, J. (2000). Repbase update: a database and an electronic journal of repetitive elements. *Trends Genet.* 16, 418–420.
- Kautz, W. H. and R. C. Singleton (1964). Nonrandom binary superimposed codes. *IEEE Trans. Inform. Theory IT-10*, 363–377.

- Kent, W. J. (2002). BLAT — the BLAST-like alignment tool. *Genome Res.* 12, 656–664.
- Lovász, L. (1993). *Combinatorial Problems and Exercises* (2nd ed.). Amsterdam: North-Holland. Problem 10.36.
- Macula, A. J. (1999). Probabilistic nonadaptive group testing in the presence of errors and DNA library screening. *Ann. Combin.* 3, 61–69.
- MacWilliams, F. J. and N. J. A. Sloane (1977). *The Theory of Error-Correcting Codes*. Amsterdam: North-Holland.
- Milosavljevic, A. (1999). DNA sequence similarity recognition by hybridization to short oligomers. U. S. patent 6,001,562.
- Ngo, H. Q. and D.-Z. Du (2002). New constructions of non-adaptive and error-tolerance pooling designs. *Discrete Math.* 243, 161–170.
- Raghavarao, D. (1971). *Constructions and Combinatorial Problems in Design of Experiments*. New York: Wiley & Sons.
- Reiman, I. (1958). Über ein Problem von K. Zarankiewicz. *Acta Math. Sci. Hung.* 9, 269–279.
- Rényi, A. (1970). *Foundations of Probability*. San Francisco: Holden-Day.
- Ruszinkó, M. (1994). On the upper bound of the size of the r -cover-free families. *J. Combin. Theory Ser. A* 66, 302–310.
- Schuler, G. D. (1997). Pieces of the puzzle: expressed sequence tags and the catalog of human genes. *J. Mol. Med.* 75, 694–698.
- Thomas, J. W., A. B. Prasad, T. J. Summers, S.-Q. Lee-Lin, V. V. B. Maduro, J. R. Idol, J. F. Ryan, P. J. Thomas, J. C. McDowell, and E. D. Green (2002). Parallel construction of orthologous sequence-ready clone-contig maps in multiple species. *Genome Res.* 12, 1277–1285.
- Velculescu, V. E., B. Vogelstein, and K. W. Kinzler (2000). Analysing uncharted transcriptomes with SAGE. *Trends Genet.* 16, 423–425.
- Venter, J. C., H. O. Smith, and L. Hood (1996). A new strategy for genome sequencing. *Nature* 381(6581), 364–366.
- Yu, W., B. Andersson, K. C. Worley, D. M. Muzny, Y. Ding, W. Liu, J. Y. Ricafrente, M. A. Wentland, G. Lennon, and R. A. Gibbs (1997). Large-scale concatenation cDNA sequencing. *Genome Res.* 7, 353–358.

List of Figures

1	Pooled Genomic Indexing (PGI)	23
2	Ambiguities and false positives	24
3	Sparse arrays	25
4	Preserved rectangles	26
5	Failure probabilities	27
6	Mapping success vs. read coverage	28

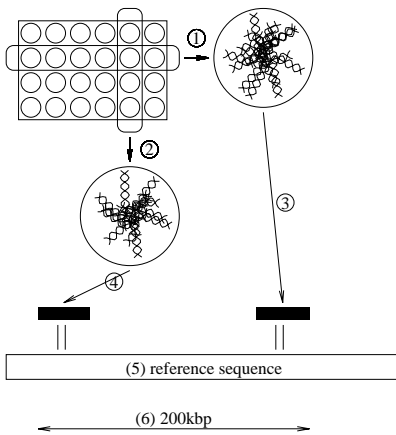


Figure 1: Pooled Genomic Indexing (PGI)

The PGI method maps arrayed clones of one species onto genomic sequence of another (5). DNA from clones are pooled together in rows (1) and columns (2). The pools are shotgun sequenced. If shotgun sequence reads from a row and a column match the reference sequence (3 and 4 respectively) within a short distance (6), the reads are assigned to the clone at the intersection of the row and the column. The clone is simultaneously mapped onto the region between the matches. The segment of the reference sequence between the two matches is said to index the clone.

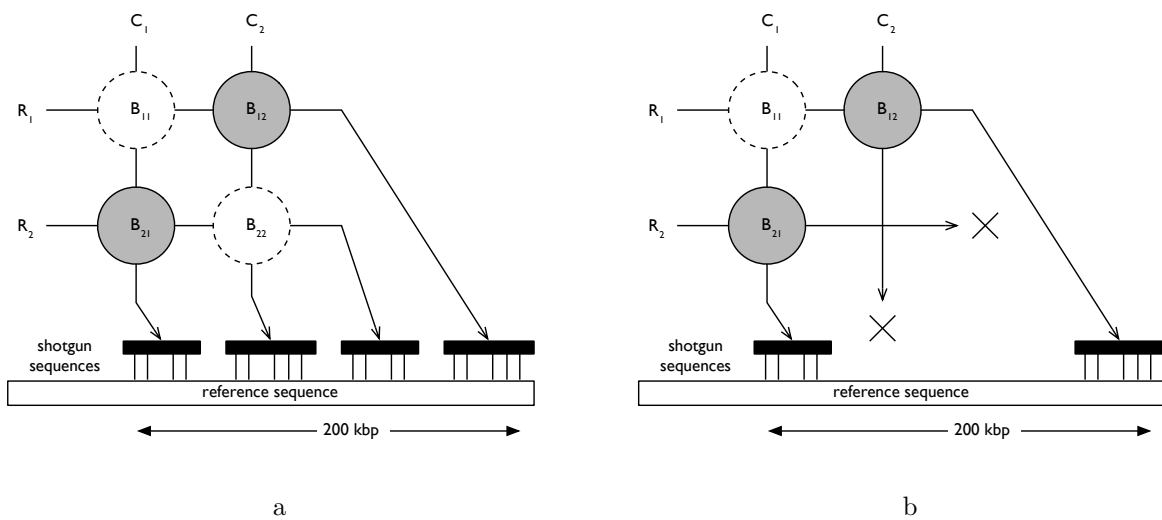
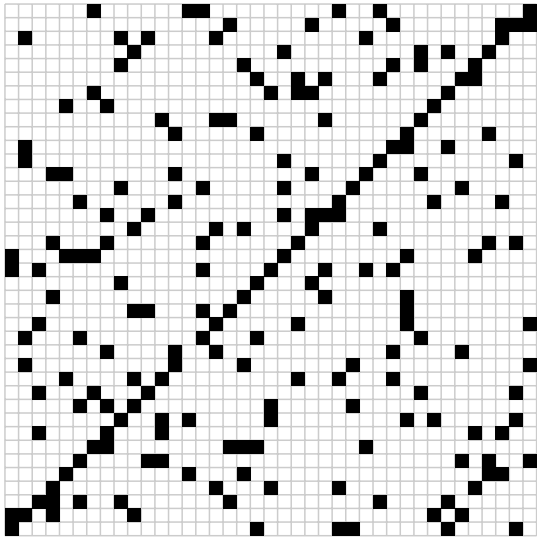
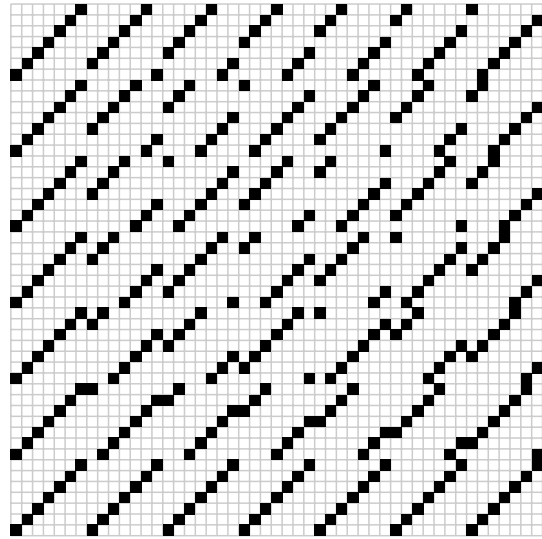


Figure 2: Ambiguities and false positives

The left-hand side shows ambiguity caused by overlap or homology between clones. Due to an overlap between clones B_{12} and B_{21} , alignments from the pools for R_1 , R_2 , C_1 , C_2 are observed within a close distance on the reference sequence. The index cannot be deconvoluted since it may originate from a homology between clones B_{11} and B_{22} . The right-hand side shows that the same overlap may cause the false positive indexing of clone B_{11} when not all pools give matching reads due to a low read coverage.



a



b

Figure 3: Sparse arrays

The left-hand side shows the sparse array used in conjunction with the mouse experiments. This 39×39 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. The right-hand side shows a 49×49 sparse array for placing 343 clones, obtained systematically using the construction described in the text.

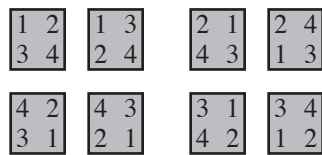


Figure 4: Preserved rectangles

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

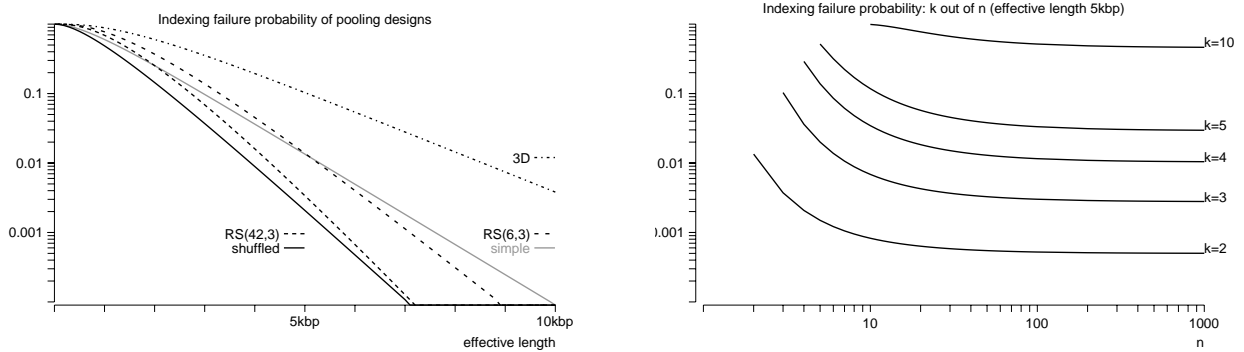


Figure 5: Failure probabilities

The graphs on the left-hand side show the probabilities for failing to find an index, as a function of the effective length. The values are calculated from Proposition 7 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, and the three-dimensional design (3D). 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 failure probabilities of pooling designs with the k -out-of- n property for an index with $cM/\ell = 10$, as given by Proposition 7. The graphs on both sides are plotted for coverage $c = 1$ and expected shotgun read length $\ell = 500$.

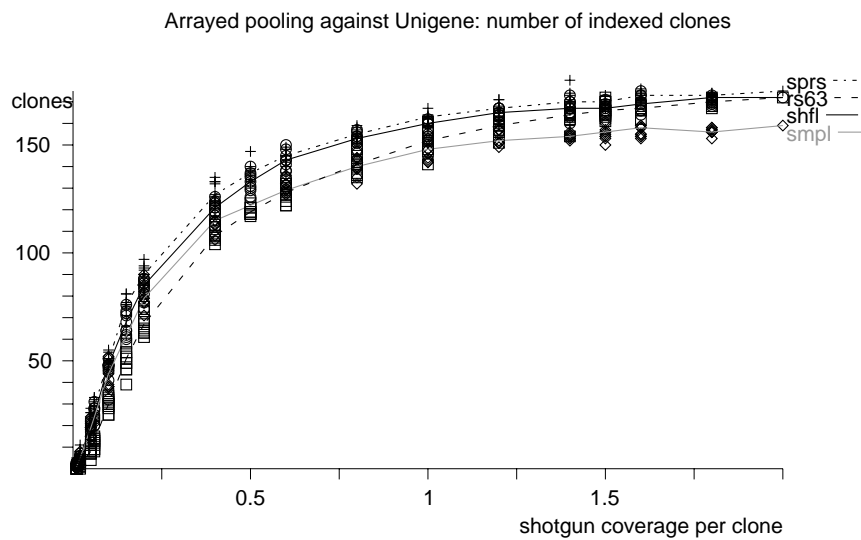


Figure 6: Mapping success vs. read coverage

Number of correctly mapped clones is indicated as a function of read 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 read coverages.

List of Tables

1	Comparison of pooling designs for 512–576 clones	30
2	Comparison of pooling designs for 9216–12167 clones	31
3	Indexing of mouse clones	32
4	Loss of information due to pooling	33
5	Indexing rat clones	34

design	N	K	n	m	amb	indexing failure probability			
						$cM/\ell = 2$	$cM/\ell = 5$	$cM/\ell = 10$	$cM/\ell = 20$
simple	576	48	2	24	1	0.60	0.16	0.013	$9.1 \cdot 10^{-5}$
sparse	512	128	2	8	2	0.60	0.16	0.013	$9.1 \cdot 10^{-5}$
double	576	96	4	24	3	0.49	0.074	$2.1 \cdot 10^{-3}$	$1.2 \cdot 10^{-6}$
triple	576	144	6	24	5	0.46	0.059	$1.2 \cdot 10^{-3}$	$3.4 \cdot 10^{-7}$
3D	512	24	3	64	1	0.88	0.47	0.10	$3.8 \cdot 10^{-3}$
RS(7, 3)	512	56	7	64	3	0.76	0.21	0.011	$1.2 \cdot 10^{-5}$
none	576	576	—	—	576	0.14	$6.7 \cdot 10^{-3}$	$4.5 \cdot 10^{-5}$	$2.1 \cdot 10^{-9}$

Table 1: Comparison of pooling designs for 512–576 clones

Columns: N is the number of clones, K is the number of shotgun libraries needed (the number of pools), n is the number of pools a clone appears in, m is the number of clones in a pool. Column **amb** gives the maximum resolvable ambiguity, i.e., the maximum number of clones an index can be deconvoluted to. Indexing failure probabilities are calculated from Equation (8). With read coverage $c = 1$ and shotgun sequence length $\ell = 500$, the corresponding effective lengths are 1000, 2500, 5000, and 10000 bp, respectively. The **simple** design is based on a 24×24 array; the **sparse** design is based on an affine plane of order 8; the **double** and **triple** designs use two and three 24×24 arrays, respectively; the **3D** design is three-dimensional pooling with a $8 \times 8 \times 8$ cube; the **RS(6, 3)** design is based on the corresponding Reed-Solomon code; **none** describes the case of preparing a shotgun library for every clone individually for the purposes of comparative mapping.

design	N	K	n	m	amb	indexing failure probability			
						$cM/\ell = 1$	$cM/\ell = 5$	$cM/\ell = 10$	$cM/\ell = 20$
simple	9216	192	2	96	1	0.60	0.16	0.013	$9.1 \cdot 10^{-5}$
sparse	12167	1058	2	23	2	0.60	0.16	0.013	$9.1 \cdot 10^{-5}$
double	9216	384	4	96	3	0.49	0.074	$2.1 \cdot 10^{-3}$	$1.2 \cdot 10^{-6}$
triple	9216	576	6	96	5	0.46	0.059	$1.2 \cdot 10^{-3}$	$3.4 \cdot 10^{-7}$
3D	9261	63	3	441	1	0.88	0.47	0.10	$3.8 \cdot 10^{-3}$
RS(22,3)	9223	506	23	401	10	0.70	0.15	$4.1 \cdot 10^{-3}$	$1.1 \cdot 10^{-6}$
none	9216	9216	—	—	9216	0.14	$6.7 \cdot 10^{-3}$	$4.5 \cdot 10^{-5}$	$2.1 \cdot 10^{-9}$

Table 2: Comparison of pooling designs for 9216–12167 clones

Columns are the same as in Table 1. The **simple** design is based on a 96×96 array; the **sparse** design is based on an affine plane of order 23; the **double** and **triple** designs use two and three 96×96 arrays, respectively; the **3D** design is three-dimensional pooling with a $21 \times 21 \times 21$ cube; the RS(22,3) design is based on the corresponding Reed-Solomon code; **none** describes the case of preparing a shotgun library for every clone individually for the purposes of comparative mapping.

	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 / 698
RS(6, 3)	42	172 (83%)	-	-	611 / 150	-	-
shuffled	58	172 (83%)	150 (72%)	180 (87%)	756 / 76	514 / 18	1549 / 238
transversal	64	181 (87 %)	-	-	748 / 100	-	-
sparse	78	175 (85%)	152 (74%)	185 (88%)	823 / 22	569 / 11	1634 / 69

Table 3: Indexing of mouse clones

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

Method	With pooling		Without pooling	
	Indexes	Indexed clones	Indexes	Indexed clones
simple	723	159 (77%)	1391	198 (96%)
RS(6, 3)	611	172 (83%)	1438	198 (96%)
shuffled	756	172 (83%)	1395	196 (95%)
transversal	748	181 (87%)	1368	195 (94%)
sparse	823	175 (85%)	1432	198 (96%)

Table 4: Loss of information due to pooling

The table lists the number of correct indexes and correctly indexed clones when indexing mouse clones with Unigene sequences.

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

Table 5: Indexing rat clones

This table show the experimental results on simulated indexing of 625 rat clones by Unigene sequences with coverage level 1.5.