

Optimization of probe coverage for high-resolution oligonucleotide aCGH

Doron Lipson^{1,*}, Zohar Yakhini^{1,2} and Yonatan Aumann³

¹Computer Science Department, Technion, Israel, ²Agilent Laboratories, CA, USA and ³Computer Science Department, Bar-Ilan University, Israel

ABSTRACT

Motivation: The resolution at which genomic alterations can be mapped by means of oligonucleotide aCGH (array-based comparative genomic hybridization) is limited by two factors: the availability of high-quality probes for the target genomic sequence and the array real-estate. Optimization of the probe selection process is required for arrays that are designed to probe specific genomic regions in very high resolution without compromising probe quality constraints.

Results: In this paper we describe a well-defined optimization problem associated with the problem of probe selection for high-resolution aCGH arrays. We propose the whenever possible ϵ -cover as a formulation that faithfully captures the requirement of probe selection problem, and provide a fast randomized algorithm that solves the optimization problem in $O(n \log n)$ time, as well as a deterministic algorithm with the same asymptotic performance. We apply the method in a typical high-definition array design scenario and demonstrate its superiority with respect to alternative approaches.

Availability: Address requests to the authors.

Contact: dlipson@cs.technion.ac.il

1 INTRODUCTION

Alterations in DNA copy number are characteristic of many cancer types and are thought to drive some cancer pathogenesis processes. These alterations include large chromosomal gains and losses as well as smaller scale amplifications and deletions. Because of their role in cancer development, regions of chromosomal instability are useful for elucidating other components of the process. For example, since genomic instability can trigger the activation of oncogenes and the silencing of tumor suppressors, mapping regions of common genomic aberrations has been used to discover cancer-related genes. Understanding genome aberrations is important for both the basic understanding of cancer and for diagnosis and clinical practice.

Alterations in DNA copy number have been initially measured using local fluorescence *in situ* hybridization-based techniques. These evolved to a genome-wide technique called Comparative Genomic Hybridization [CGH, (Kallioniemi *et al.*, 1993)], now commonly used for the identification of chromosomal alterations in cancer (Mertens *et al.*, 1997; Balsara and Testa, 2002). In this genome-wide cytogenetic method differentially labeled tumor and normal DNA are co-hybridized to normal metaphases. Ratios between the two labels allow the detection of chromosomal amplifications and deletions of regions that may harbor oncogenes and tumor suppressor genes. Classical CGH has, however, a limited

resolution (10–20 Mb). With such low resolution it is impossible to predict the borders of the chromosomal changes or to identify changes in copy numbers of single genes and small genomic regions. In a more advanced method termed array-based CGH (aCGH), tumor and normal DNA are co-hybridized to a microarray of thousands of BAC, cDNA or oligonucleotide probes (Pinkel *et al.*, 1998; Pollack *et al.*, 1999; Lucito *et al.*, 2003; Brennan *et al.*, 2004; Bignell *et al.*, 2004; Barrett *et al.*, 2004). The use of aCGH allows the determination of changes in DNA copy number of relatively small chromosomal regions. Using oligonucleotide arrays the resolution can, in theory, be finer than single genes.

In fact, when using oligonucleotides the resolution appears to have no limitation as these can be designed to probe any region in the genome of interest. In reality the resolution is limited for two main reasons. One is the fact that not all genomic locations can be effectively probed. For example, genomic sequences that are not unique (genomic repeats) or genomic regions with extreme GC content limit the design of specific probes. The other limitation is array real-estate—the number of genomic regions that can be probed in one array. Note that for technology implementations that use redundant probes this number is much smaller than the number of features on the array.

In this work we address the joint optimization of these two resolution limiting factors. The methods we develop are useful in the context of designing high-definition arrays. These are arrays designed to probe specific genomic regions in very high resolution. For this purpose a dense set of probes is selected for the region of interest. We are interested in doing so while maintaining optimal coverage and without compromising probe quality constraints. Major determinants of probe quality are its expected sensitivity, which is predicted by thermodynamic evaluation of the probe–target complex, and specificity, which is inferred from the extent to which close repeats of the probe can be found in the genome. In previous work (Lipson *et al.*, 2002; Li and Stromo, 2001; Mei *et al.*, 2003; Rouillard *et al.*, 2003) probe sensitivity and specificity have been studied mostly in the context of expression profiling. The background context in CGH is different and so is the thermodynamics but the overall considerations remain valid. The methods described in the current paper are completely independent from those used for assigning quality to candidate probes and can therefore be applicable to any such assignment, including one derived from experimental data.

Lucito *et al.* (2003) describe design criteria that are based only on an empirically derived quality assignment. That is, they choose to use all probes with a given quality or better without taking any uniformity considerations into account. This approach leads to having regions that are more densely probed than others as well as to

*To whom correspondence should be addressed.

coverage discontinuities. When studying a given genomic region we are typically not biased to any specific subregions and seek to obtain copy number information for all subregions. It is therefore best to have a uniform (equidistant) distribution of probes in the region. This way we best avoid coverage discontinuities and have comparable information about all subregions. However, a uniformly distributed set of probes will greatly compromise quality considerations. We may be selecting probes that are highly non-specific, for example. This paper describes methods that assume a given quality threshold above which probes are acceptable and then optimize the coverage using only these candidate probes and working with the array real-estate constraints.

The rest of the paper is organized as follows: in Section 2 we describe a new score that accounts for probe coverage, and formally define the optimization problem. In Section 3 we describe two algorithms that efficiently solve the defined problem: a fast stochastic algorithm, and a deterministic variant. Finally, in Section 4, we demonstrate the application of our method in a biological scenario, and compare the obtained results with some alternative methods.

2 MATHEMATICAL FORMULATION

In this section we formally define the optimization problems associated with probe selection for aCGH arrays, as discussed in the Introduction. We believe that such a formal definition is of independent interest, as the problem is most commonly stated in ill-defined terms.

Two types of parameters are typically considered when evaluating candidate probes for a hybridization assay. Sensitivity is a measure of a probe's ability to strongly interact with its target, and is typically assessed by considering the thermodynamic stability of the probe–target complex. Specificity is a measure of a probe's ability to discriminate between its intended target and other non-specific molecules it might cross hybridize to, and is typically assessed by considering the similarity of the target to the expected molecular background (e.g. the entire transcriptome in an expression profiling assay, or the entire genome in an aCGH assay). Methods for predicting the sensitivity and specificity of candidate probes have been extensively studied in the past (see Introduction) and will not be considered here. Instead, we shall assume we are provided with a quality parameter $q(p)$, specifying the overall predicted performance of the probe p .

When designing oligonucleotide probes for aCGH an additional criterion should also be taken into consideration—that of coverage. Specifically, when an array is designed for the purpose of pinpointing genomic breakpoints, an important consideration in the design is to minimize the uncertainty at which the breakpoints are mapped, wherever they may be. Thus, probes should be somehow uniformly spaced, so as to be not 'too far' from the location of any possible breakpoint.

Accordingly, when designing an aCGH array, we are interested in selecting 'highest quality' probes, with the 'best possible' coverage. In this section we provide formal definitions that capture the intuitive meaning of these notions.

2.1 Probe quality

Let P be the set of candidate probes. For brevity, we identify each probe with its genomic location, thus $P \subset \mathbb{N}$. Let $q : P \rightarrow \mathbb{R}^+$ be the quality function associating a quality score with each probe.

Intuitively, we are interested in using the 'best possible' probes, i.e. those with the highest quality score. However, the high quality probes may not be evenly distributed within the genomic area of interest. Thus, choosing the globally best probes may result in poor coverage of the genome. Hence, rather than using the globally highest quality probes, we seek to use the locally best probes. Specifically, for a probe $p \in P$ and window size w , we define the w -local quality of p to be the *percentile* of $q(p)$ within scores in the w window around p . Formally, let P_w be the set of candidate probes within the window $[p - w/2, p + w/2]$. Then

$$q_w(p) = \frac{|\{p' \in P_w : q(p') \leq q(p)\}|}{|P_w|} \quad (1)$$

We now define the 'good probes' to be those with high local quality. Formally, for a window size w and threshold $\tau \in [0, 1]$, the w -local τ -good probes are all the probes p for which $q_w(p) \geq \tau$. We shall seek to use only probes which are w -local τ -good, for some proper choice of w and τ . We note that it is also possible to optimize for τ , as explained later.

2.2 Probe coverage

As mentioned, we are also interested that the probes 'cover' the genomic region of interest with the most granular coverage possible. Specifically, if there is a breakpoint at some point on the genome, we would like to be able to determine its location as precisely as possible. However, two factors limit the precision that can be obtained:

- With a limited number of probes in the array, probes cannot be placed at each genomic location. Rather, they must be spread across the region of interest. In this case, the localization of genomic events can only be determined by the pair of flanking probes. We shall seek that for all genomic locations, the gap between this pair of probes is as small as possible.
- Some genomic regions do not contain any candidate probes, or only probes of poor quality. In such cases, large gaps between probes are inevitable. Any breakpoint occurring within these gaps can only possibly be localized to within the pair of candidate probes bordering the gap.

Thus, we seek to choose a set of probes that uniformly cover all genomic locations—whenever possible, and as close as possible to the points within the large gaps—otherwise. The following definition captures this intuition:

DEFINITION 2.1. *Given a genomic region G , a set of candidate probes P and a parameter ϵ , a subset $C = (c_1, \dots, c_k) \subseteq P$ is a whenever possible (WP) ϵ -cover of G with respect to P if for any genomic location $x \in G$, the following holds. Let c_i and c_{i+1} be the two selected probes closest to x from the left and from the right, respectively (if $x < c_1$ then c_0 is set to be the left-end of G , and for $x > c_k$, c_{k+1} is the right-end of G). Then, one of the following holds:*

- (1) $c_{i+1} - c_i \leq \epsilon$ (i.e. the flanking selected probes are within ϵ distance of each other), or
- (2) there is no candidate probe between c_i and c_{i+1} .

For such a cover C , we say that the resolution of C is ϵ .

Thus, a WP ϵ -cover of G guarantees that for any possible breakpoint x , it can either be localized to within ϵ base pairs, or to within the best resolution that could have been obtained even if all candidate probes would have been used. We believe that the notion of WP ϵ -cover faithfully captures the requirements of probe selection.

2.3 The optimization problem

Given a fixed number of probes in the array, the problem of designing an aCGH array is therefore a bicriteria optimization problem: select a subset of probes that are (i) of high quality (ii) with high resolution. A standard approach to such bicriteria problems is to optimize one criterion, given a constraint on the other. In our case, this gives rise to the following two optimization problems:

PROBLEM 2.2. (Probe selection—resolution optimization). *Given an integer k , genomic region G , window size w and quality threshold τ , find the minimal possible resolution ϵ^* and a probe subset $C \subset P$ such that:*

- (1) $|C| = k$,
- (2) C contains only τ -good probes,
- (3) C is a WP ϵ^* -cover of G with respect to the τ -good probes.

PROBLEM 2.3. (Probe selection—quality optimization). *Given an integer k , genomic region G , window size w and resolution threshold ϵ , find a maximum quality score τ^* , and a probe subset $C \subset P$ such that:*

- (1) $|C| = k$,
- (2) C contains only τ^* -good probes,
- (3) C is a WP ϵ -cover of G with respect to the τ^* -good probes.

In this paper we focus on the resolution optimization version, but also provide an efficient solution to the quality optimization version.

2.4 Problem variants

2.4.1 Multiple genomic segments A simple but important variant of the problem involves multiple genomic segments. In many cases, an aCGH array is designed to assay more than a single genomic segment at a time, most typically different chromosomal segments. In this case we would like to uniformly cover all the genomic regions of interest.

2.4.2 Biased selection Another useful variant of the problem involves biased selection of probes in certain genomic segments. The most typical application of this variant is for increasing the resolution of probes in gene coding regions at the expense of non-coding regions, while preserving the uniform resolution within each of the subtypes.

3 ALGORITHMS

In this section we describe efficient algorithms for the probe selection problem, focusing on the resolution optimization version. We first show how to determine the minimal number of probes required for a WP ϵ -cover, for a given ϵ . We then use this procedure to search for the minimal ϵ for a given number of probes. We provide a fast randomized algorithm, described in Section 3.3.1, that finds the optimal ϵ in $O(n \log n)$ steps. A deterministic algorithm with the

same asymptotic performance is described in Section 3.3.2. Finally, we show how these algorithms can be extended to accommodate the variants of the problem that were described in Section 2.4.

3.1 Determining the τ -good probes

Given a window size w , it is possible to calculate $q_w(p)$ for each $p \in P$ by scanning the genome with a sliding window of size w . As we slide the window from left to right, at each step, at most one candidate probe is added and one omitted. We maintain the quality scores of the probes in the window in a balanced binary search tree. This tree can be maintained in $O(n \log m)$ steps, using any of the known balanced tree data structures (here, m is the maximum number of probes in a window of size w , $n = |P|$). For each probe p , its w -local quality score can be obtained in $O(n \log m)$ steps, by finding its rank in the search tree. Thus, the w -local quality of all candidate probes can be determined in $O(n \log m)$ steps. This also allows to determine the τ -good probes, for any threshold τ . From here and on, we denote by $\bar{P} = (p_1, p_2, \dots, p_n) (p_1 < p_2 < \dots < p_n)$ the sequence of τ -good candidate probes, for the chosen threshold τ .

3.2 Finding the minimal size for a WP ϵ -cover

In order to solve the resolution optimization problem, we first consider the reverse optimization problem:

PROBLEM 3.1. *Given a genomic region $G = [g_{\text{beg}}, g_{\text{end}}]$ and a fixed value of ϵ , find a WP ϵ -cover C for G of minimal size.*

Algorithm 1, based on a greedy approach, solves this problem in $O(\bar{n})$ steps.

Algorithm 1 Find a minimal size WP ϵ -cover

FindMinCover(ϵ)

- 1: $c_0 \leftarrow g_{\text{beg}}$
 - 2: $i \leftarrow 0$
 - 3: **While** $(g_{\text{end}} - c_i) > \epsilon$ **do**
 - 4: Set c_{i+1} to be the rightmost candidate probe following c_i such that $(c_{i+1} - c_i) \leq \epsilon$.
 - 5: If no such probe exists: c_{i+1} is the next candidate probe to the right of c_i .
 - 6: $i \leftarrow i + 1$
 - 7: **return** $C = \{c_1, \dots, c_i\}$.
-

CLAIM 3.2. *The subset C returned by Algorithm 1 is a WP ϵ -cover of G .*

PROOF. Consider a point $x \in G$. Let c_i be leftmost probe in C to the right of x . Consider two cases:

- (1) c_i is within distance ϵ of c_{i-1} .
- (2) c_i is not within distance ϵ of c_{i-1} . In this case, by the algorithm, c_{i-1} and c_i are consecutive candidate probes in \bar{P} .

CLAIM 3.3. *There is no WP ϵ -cover C^* for G with $|C^*| < |C|$.*

PROOF. Assume there is such a WP ϵ -cover C^* , with $|C^*| = k^*$. Let $c_i^* \in C^*$ be the first i such that $c_i^* > c_i$ (there must be one since $c_k^* > c_k$). Accordingly, $c_{i-1}^* \leq c_{i-1}$. Two cases are possible:

- (1) $(c_i - c_{i-1}) > \epsilon$ and therefore $(c_i^* - c_{i-1}^*) > \epsilon$.
- (2) c_i was chosen as the rightmost probe following c_{i-1} such that $(c_i - c_{i-1}) \leq \epsilon$. Consequently, $(c_i^* - c_{i-1}^*) > \epsilon$.

Let x be a genomic location strictly between the two selected probes c_{i-1}^* and c_i^* . $(c_i^* - c_{i-1}^*) > \epsilon$ and there exists an unselected candidate probe c_i between c_{i-1}^* and c_i^* contradicting the fact that C^* is a WP ϵ -cover.

3.3 Optimizing resolution

In the probe selection problem, we are given a number k of probes and seek to optimize the resolution. We do so by performing a binary search on the resolutions, using Algorithm 1 as the decision criteria. The recursive binary search procedure is described in Algorithm 2.

Algorithm 2 Find a k -sized WP ϵ -cover with minimal ϵ .

UniProbe (k, G, \bar{P})

1: return RecursiveOptimizeResolution ($k, 1, |G|$)

RecursiveOptimizeResolution ($k, \text{low}, \text{high}$)

1: **if** low = high **then**

2: $C \leftarrow \text{FindMinCover}(\text{low})$

3: **return** C

4: split $\leftarrow \text{FindSplit}(\text{low}, \text{high})$

5: $C \leftarrow \text{FindMinCover}(\text{split})$

6: **if** $|C| > k$ **then**

7: **return** RecursiveOptimizeResolution ($k, \text{split}, \text{high}$)

8: **else**

9: **return** RecursiveOptimizeResolution ($k, \text{low}, \text{split}$)

The procedure performs a binary search for the optimal ϵ within the range $[1, |G|]$. Clearly, the optimal ϵ is a distance between some two probes $p_i, p_j \in \bar{P}$ (otherwise, ϵ could be reduced to the closest distance). There are $O(\bar{n}^2)$ such pairwise distances. Thus a balanced binary search could pinpoint the optimal value in $O(\log \bar{n})$ recursive calls. However, these $O(\bar{n}^2)$ pairwise distances are not provided to us explicitly, and enumerating them all would take $O(\bar{n}^2)$ steps by itself. Thus, we seek to somehow perform a balanced binary search on this $O(\bar{n}^2)$ -sized space without explicitly enumerating it. Interestingly, this can be done in $O(\bar{n} \log \bar{n})$ time, as explained hereunder. The key element is to find a good split value (Line 4) in linear time. We provide both randomized and deterministic procedures to find such a split value efficiently.

Each invocation of the RecursiveOptimizeResolution procedure takes $O(\bar{n})$ steps (the complexity of FindMinCover). Provided that the depth of the recursion is $O(\log \bar{n})$, the overall complexity is $O(\bar{n} \log \bar{n})$.

3.3.1 Fast randomized split selection We first describe a fast randomized procedure for the selection of the split value. The procedure provides that the expected number of recursive calls in Algorithm 2 is $O(\log \bar{n})$. A description of the algorithm is provided as Algorithm 3, and an explanation follows.

Algorithm 3 Fast randomized selection of split value

Randomized-FindSplit (low, high)

1: **for** $i = 1, 2, \dots, \bar{n}$ **do**

2: $\ell(i) \leftarrow \text{argmin}\{p_j: |p_j - p_i| \in [\text{low}, \text{high}]\}$

3: $h(i) \leftarrow \text{argmax}\{p_j: |p_j - p_i| \in [\text{low}, \text{high}]\}$

4: $\sigma \leftarrow \sum_{i=1}^{\bar{n}} h(i) - \ell(i) + 1$

5: $r \leftarrow$ random integer in $[1, \sigma]$

6: $\langle i_0, j_0 \rangle \leftarrow r$ -th element of the set

$\{(i, j) \mid i \in [1.. \bar{n}], j \in [\ell(i).. h(i)]\}$

7: **return** split = $|p_{i_0} - p_{j_0}|$

Let $X = X(\text{low}, \text{high})$ be the set of all pairs of probes $\langle p_i, p_j \rangle$ such that the distance between the two is within the range $[\text{low}, \text{high}]$. We wish to find a split value split such that the number of pairs in X for which the distance is above split is roughly the same as the number of pairs for which the distance is under split. For a given starting probe p_i , let $\ell(i)$ be the smallest index such that $\langle p_i, p_{\ell(i)} \rangle \in X$ (i.e. $|p_{\ell(i)} - p_i| \in [\text{low}, \text{high}]$), and let $h(i)$ be the largest such index. Clearly, for any j between $\ell(i)$ and $h(i)$, $\langle p_i, p_j \rangle \in X$. The algorithm first determines $\ell(i)$ and $h(i)$, for all $i = 1, \dots, \bar{n}$. This can be completed in $O(\bar{n})$ steps, as explained later. Given these values, we can compute the size of X (Line 4), and choose a random element from this set (Lines 5–6). The distance of this pair is the split value (Line 7). It is easy to see that with probability half the split value is between the 25-th and 75-th percentile of the distances of pairs in X . Hence, after an expected $O(\log \bar{n})$ recursive calls of Algorithm 2, the recursion ends.

It remains to show how to efficiently compute $\ell(i)$ and $h(i)$. Note that $\ell(1) \leq \ell(2) \leq \dots \leq \ell(\bar{n})$, and similarly for $h(i)$. Thus, with a single scan over the indexes $1, \dots, \bar{n}$, we can determine all $\ell(i)$ s and $h(i)$ s. We thus obtain:

CLAIM 3.4. Algorithms 2 and 3 solve the resolution optimization version of the probe selection problem in $O(\bar{n} \log \bar{n})$ expected time (where \bar{n} is the number of τ -good candidate probes).

3.3.2 Deterministic split selection A deterministic procedure for split selection, which also runs in $O(\bar{n})$ time, can be obtained using the methods for selection in sorted matrices, directed sum matrices, point distances in \mathbb{R}^d , and other multisets (6,17,1). A full description of the algorithm, as it applies to our setting, is provided in the appendix. With this algorithm we obtain:

CLAIM 3.5. The resolution optimization version of the probe selection problem can be solved deterministically in $O(\bar{n} \log \bar{n})$ steps (where \bar{n} is the number of τ -good candidate probes).

In practice, the randomized algorithm is substantially simpler and faster, and provides the same results.

3.4 Algorithmic variants

We now show how to solve the different problem variants discussed in Sections 2.3 and 2.4.

3.4.1 Quality optimization Suppose we wish to optimize quality, rather than resolution, as described in Problem 3. In this case we perform an algorithm similar to Algorithm 2, using a binary search on quality, rather than resolution. In this case the number of different quality values is bounded by n , so a simple binary search can be employed.

3.4.2 Multiple genomic segments The case of multiple genomic regions is handled almost identically to the single segment case. To do so, note that for any collection of disjoint genomic segments G_1, G_2, \dots, G_t , any WP ϵ -cover C for this collection can be divided into a collection of disjoint covers C_1, C_2, \dots, C_t , each constituting a WP ϵ -cover for the corresponding G_j . Thus, the minimal WP ϵ -cover for G_1, G_2, \dots, G_t , is the union of the minimal covers for each of the G_j s. Accordingly, in the binary search algorithm (Algorithm 2), we create the minimal cover for the collection of segments (Lines 2 and 5) as the union of the individual covers. All

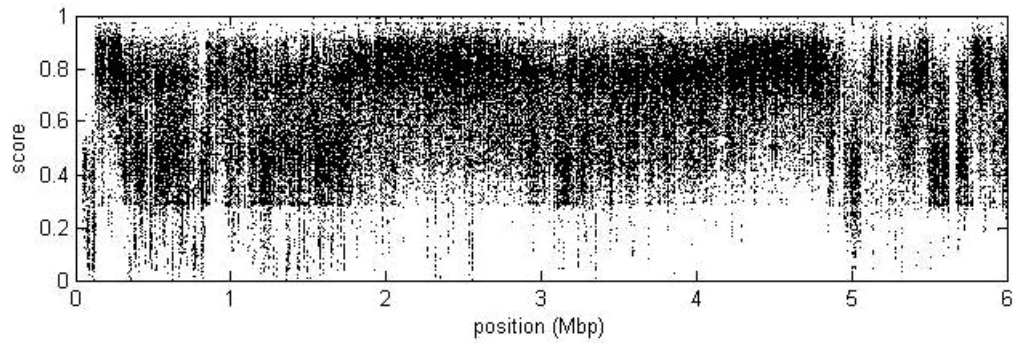


Fig. 1. The 82 500 candidate probes in chromosome 10 0–6 Mb. Each point represents the genomic position and predicted quality score of a single probe.

the rest of the algorithms remain the same, with the understanding that the distance between probes in separate segments is infinite.

3.4.3 Biased selection Suppose that we have two types of regions, say coding and non-coding, and we seek a cover for which the resolution in the coding regions is γ more granular than that in the non-coding regions. To do so, we multiply all distances within the coding regions by a factor of γ and run the algorithm described above. We will obtain an optimal cover, such that the resolution of the coding region is smaller by a factor of γ from that of the non-coding region. This technique can be extended for more than two types of regions as well.

4 APPLICATION

We demonstrate the application of UniProbe (Algorithm 2) on a typical scenario of high-resolution probe design. Assume we are searching for a genomic breakpoint at the 6 Mb p-terminus of chromosome 10. Given a set of candidate probes, we would like to select 3000 oligonucleotide probes that offer the best guaranteed resolution of detecting breakpoints. Figure 1 depicts a set of 82 500 candidate probes, taken from a probe database (unpublished data). Each point represents the genomic position and quality score of a single probe, predicted from thermodynamical consideration of the probe sequence and comparison to the entire genome background.

In theory, placing the probes at equal spacing would provide a guaranteed 2 Kb resolution, in the sense that the genomic distance between each pair of consecutive probes would be 2 Kb. However, achieving this theoretical resolution is impossible. For example, unclosed gaps in the genomic sequence at positions 0–50 Kb and 5.63–5.68 Mb [UCSC Genome Browser, Kent *et al.*, (2002)] are genomic regions at which maximal resolution cannot be obtained. In addition, we require that highest quality probes are used while sustaining a high degree of uniformity. As can be seen in Figure 1 this dual objective may force us to use probes of lower quality in genomic regions that contain only inferior probes, e.g. at 50–130 Kb or 4.97–5.07 Mb.

We compare the set of probes selected by UniProbe to two different naive methods for uniformly spaced probe selection. The first method (Qual) selects probes on the basis of their quality alone, assuming that randomness in the positions of the superior probes will lead to some degree of uniformity in their coverage. This method was described by Lucito *et al.* (2003) for designing a whole-genome representational oligonucleotide array, where the

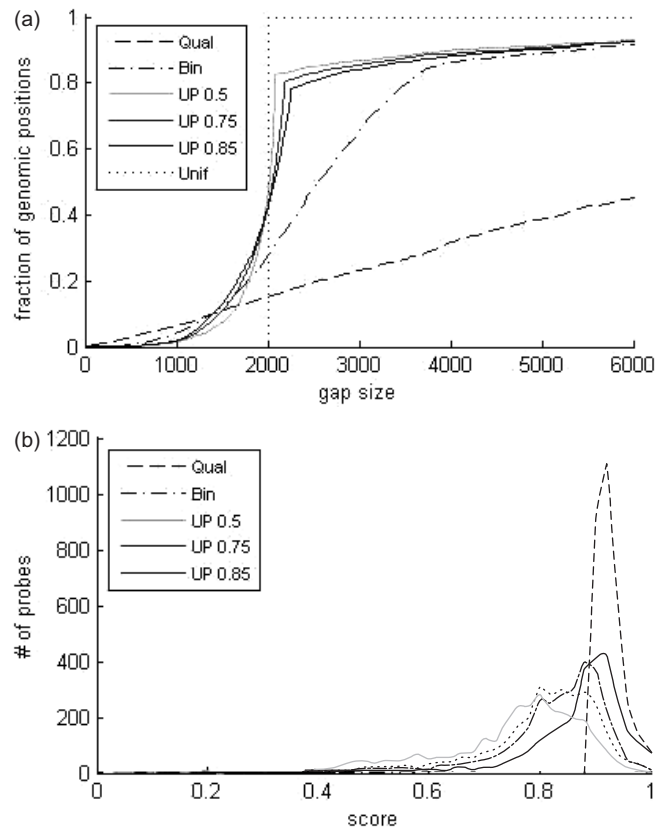


Fig. 2. Comparison of the resolution performance of the naive algorithms Qual and Bin, and the new algorithm UniProbe (UP) with quality threshold $\tau = 0.5, 0.75, 0.85$. (a) Resolution performance—the curve for each algorithm describes the fraction of genomic positions within the target genomic segment that lie within a gap of up to the given size between adjacent selected probes. Unif describes the theoretical optimal result that could be expected from probes that are placed at equal spacing along the segment. (b) Quality performance—the curve for each algorithm describes the distribution of quality scores of the selected probes.

probes with the best empirical performance were chosen. Here, we select the 3000 probes with highest quality score. The second naive method (Bin) is based on binning: the total genomic segment is divided into 3000 equally sized bins, and the highest quality probe in each bin is selected. Note that no probes can be selected from

empty bins, reducing the total number of selected probes to below 3000.

Figure 2a compares of the resolution performance of the different algorithms—Qual, Bin and UniProbe with three different values of τ : 0.5, 0.75 and 0.85 ($w = 2$ Kb). For each algorithm, and for each genomic distance $d \leq 6$ Kb, we note the fraction of genomic positions in the complete segment that lie within gaps of size $\leq d$ between adjacent selected probe. For comparison we depict the optimal performance that could be expected from probes that are placed at equal spacing along the segment (Unif). Figure 2b depicts the quality distributions of the selected probes.

It is clear that the algorithm Qual performs very badly, in terms of resolution, as could be expected from an algorithm that does not take uniformity directly into consideration, although the probe qualities are clearly superior. The algorithm UniProbe guarantees a ‘whenever possible’ resolution of $\epsilon = 2085,2186,2258$ for $\tau = 0.5, 0.75, 0.85$, respectively, and some 80% of all genomic positions are located within gaps of size $\leq \epsilon$ between adjacent selected probes. The remaining 20% are located within gaps that cannot be probed at the guaranteed resolution. The new algorithm outperforms Bin for all three values of τ . The performance of Bin converges with UniProbe for values of $d > 4000$. This observation is explained by the fact that Bin may deviate from the optimal solution by a maximal factor of 2 (the maximal distance between probes in two consecutive non-empty bins is twice the size of the bin). Overall, the resolution obtained by UniProbe is close to optimal, with values of ϵ approaching the optimal 2 Kb. By definition, this resolution is guaranteed for all genomic locations for which this is possible whereas the closest possible probes are guaranteed for the remaining locations, which are located within gaps in the candidate probe set. The quality of the probes selected by UniProbe, although slightly inferior to those selected by Bin, are still satisfactory high.

REFERENCES

- Agarwal,P.K. et al. (1993) Selecting distances in the plane. *Algorithmica*, **9**, 495–514.
- Balsara,B.R. and Testa,J.R. (2002) Chromosomal imbalances in human lung cancer. *Oncogene*, **21**, 6877–6883.
- Barrett,M.T. et al. (2004) Comparative genomic hybridization using oligonucleotide microarrays and total genomic DNA. *PNAS*, **101**, 17765–17770.
- Bignell,G.R. et al. (2004) High-resolution analysis of DNA copy number using oligonucleotide microarrays. *Genome Res.*, **14**, 287–295.
- Brennan,C. et al. (2004) High-resolution global profiling of genomic alterations with long oligonucleotide microarray. *Cancer Res.*, **64**, 4744–8.
- Johnson,D.B. and Mizoguchi,T. (1978) Selecting the k th element in $x + y$ and $x_1 + x_2 + \dots + x_m$. *SIAM J. Comput.*, **7**, 147–153.
- Kallioniemi,O.P. et al. (1993) Comparative genomic hybridization: a rapid new method for detecting and mapping DNA amplification in tumors. *Semin Cancer Biol.*, **4**, 41–46.
- Kent,W.J. et al. (2002) The human genome browser at UCSC. *Genome Res.*, **12**, 996–1006.
- Li,F. and Stormo,G.D. (2002) Selection of optimal DNA oligos for gene expression arrays. *Bioinformatics*, **17**, 1067–1076.
- Lipson,D. et al. (2002) Designing specific oligonucleotide probes for the entire *S. cerevisiae* transcriptome. In Second Workshop on Algorithms in Bioinformatics (WABI 02). *LNCS*, **2452**, 491–505.
- Lucito,R. et al. (2003) Representational oligonucleotide microarray analysis: a high-resolution method to detect genome copy number variation. *Genome Res.*, **13**, 2291–2305.
- Mei,R. et al. (2002) Probe selection for high-density oligonucleotide arrays. *PNAS*, **100**, 11237–11242.
- Mertens,F. et al. (2002) Chromosomal imbalance maps of malignant solid tumors: a cytogenetic survey of 3185 neoplasms. *Cancer Res.*, **57**, 2765–2780.
- Pinkel,D. et al. (1998) High resolution analysis of DNA copy number variation using comparative genomic hybridization to microarrays. *Nat. Genet.*, **20**, 207–211.
- Pollack,J.R. et al. (1999) Genome-wide analysis of DNA copy-number changes using cDNA microarrays. *Nat. Genet.*, **23**, 41–46.
- Rouillard,J. et al. (2003) Oligoarray 2.0: design of oligonucleotide probes for DNA microarrays using a thermodynamic approach. *Nucleic Acids Res.*, **31**, 3057–3062.
- Salowe,J.S. (1989) L-infinity interdistance selection by parametric search. *Inform. Process. Lett.*, **30**, 9–14.

APPENDIX

Here, we show how to deterministically find a split point in $O(\bar{n})$. As noted, the algorithm follows the ideas of (Johnson and Mizoguchi, 1978; Salowe, 1989; Agarwal *et al.*, 1993) for selection in sorted matrices, direct sum sets, point distances in \mathbb{R}^d , and other multisets. We describe the algorithm as it pertains to our problem. The algorithm provides that we find a split value that is guaranteed to be ‘relatively-balanced’, i.e. for at least one-fourth of pairs in X the distance is above the split value, and for at least one-fourth of the pairs—the distance is below (recall that X is the set of all probe pairs $\langle p_i, p_j \rangle$ such that the distance between the two is in the range [low, high]). A pseudocode description of the procedure is provided as Algorithm 4, and an explanation follows.

In the procedure, we use the following notations. For each $i = 1, \dots, \bar{n}$ we denote:

- $\ell(i)$ —smallest index such that $|p_{\ell(i)} - p_i| \in [\text{low}, \text{high}]$
- $h(i)$ —largest index such that $|p_{h(i)} - p_i| \in [\text{low}, \text{high}]$
- $d(i) = h(i) - \ell(i) + 1$ —number of pairwise distances within the range [low, high] that start at the probe p_i
- $m(i) = \lfloor \frac{\ell(i) + h(i)}{2} \rfloor$ —midpoint between $\ell(i)$ and $h(i)$
- $v(i) = |p_{m(i)} - p_i|$ —distance from p_i to its corresponding midpoint.

The values of $\ell(i)$ and $h(i)$ can be computed in $O(\bar{n})$ steps for all i collectively, as explained above in the description of the randomized procedure. The other values can be computed from $\ell(i)$ and $h(i)$ in an additional $O(\bar{n})$ steps.

Algorithm 4 Deterministic selection of split value

Deterministic-FindSplit (low,high)

```

1: for all  $i = 1, \dots, \bar{n}$  do
2:   initialize  $d(i), m(i), v(i)$ 
3:  $\sigma \leftarrow \sum_{i=1}^{\bar{n}} d(i)$ 
4:  $S \leftarrow \{1, \dots, \bar{n}\}$ 
5: return RecursiveSplit ( $S, 0$ )
RecursiveSplit ( $S, b$ )
1: if  $|S| = 1$  then
2:   let  $i_0$  be such that  $S = \{i_0\}$ 
3:   return  $v(i_0)$ 
4:  $\alpha \leftarrow \text{median of } \{v(i) : i \in S\}$ 
5:  $S^- \leftarrow \{i \in S : v(i) \leq \alpha\}$ 
6:  $S^+ \leftarrow \{i \in S : v(i) \geq \alpha\}$ 
7:  $\hat{b} \rightarrow \sum_{i \in S^-} \lceil d(i)/2 \rceil$ 
8: if  $\hat{b} + b > \sigma/4$  then
9:   return RecursiveSplit ( $S^-, b$ )
10: else
11: return RecursiveSplit ( $S^+, b + \hat{b}$ )

```

Consider the set of mid-point distances $\{v(i) : i \in [1, \dots, \bar{n}]\}$. For a given \hat{i} , let

$$\text{below}(\hat{i}) = \{\langle i, j \rangle : v(i) \leq v(\hat{i}), j \leq m(i)\} \quad (2)$$

For any $\langle i, j \rangle \in \text{below}(\hat{i})$, necessarily $|p_j - p_i| \leq v(\hat{i})$. Similarly, setting

$$\text{above}(\hat{i}) = \{\langle i, j \rangle : v(i) \geq v(\hat{i}), j \geq m(i)\} \quad (3)$$

we have that $|p_j - p_i| \geq v(\hat{i})$ for all $\langle i, j \rangle \in \text{above}(\hat{i})$. Thus, we seek to find an i for which both below (i) and above (i) are large. Specifically, let i^* be such that:

- (1) $|\text{below}(i^*)| \geq |X|/4$,
- (2) below(i^*) is the smallest possible, subject to the above constraint.

Then,

CLAIM 0.1. For i^* as defined above,

- (1) $|\text{below}(i^*)| \geq |X|/4$
- (2) $|\text{above}(i^*)| \geq |X|/4$

PROOF.

- (1) By definition $|\text{below}(i^*)| \geq |X|/4$.
- (2) Suppose that $|\text{above}(i^*)| < |X|/4$

Let $i_1 = \text{argmax}\{v(i) : v(i) < v(i^*)\}$. Note that $\text{below}(i_1) \cup \text{above}(i^*)$ covers at least half of X . Thus, $|\text{below}(i_1)| \geq |X|/4$. However, $\text{below}(i_1) \subset \text{below}(i^*)$, in contradiction to the minimality of below(i^*).

Thus, choosing $v(i^*)$ as a split value guarantees that at least a constant fraction of the search space is eliminated.

CLAIM 0.2. Algorithm 4 returns $v(i^*)$ in $O(\bar{n})$ steps.

PROOF. We first prove that it returns $v(i^*)$. By induction we prove that at all recursive calls to the function RecursiveSearch, S always contains i^* . Initially, S is $\{1, \dots, \bar{n}\}$ and the claim holds. Suppose that $i^* \in S$ for some call of the recursive function. Consider the i for which $v(i)$ is the median chosen in Line 4. If $i^* < i$ then $i^* \in S^-$. The value of $b + \hat{b}$ is exactly the size of below(i). Hence, the test at Line 8 will succeed and the next iteration will be with $S = S^-$ and the claim holds. Similarly, if $i^* \geq i$ then $i^* \in S^+$. Now the test at Line 8 will fail and the next iteration will be with $S = S^+$, and the claim holds.

The running time of executing a single iteration of the function RecursiveSearch is dominated by Lines 4–7, all of which can be completed in $O(|S|)$ steps. At each recursive call, the size of S is reduced by half. Hence, the entire process is completed in $O(\bar{n})$ steps.