Nucleic Acids Research, 1993, Vol. 21, No. 8 1965–1974

Algorithms and software tools for ordering clone libraries: application to the mapping of the genome of *Schizosaccharomyces pombe*

Richard Mott, Andrei Grigoriev, Elmar Maier, Jorg Hoheisel and Hans Lehrach
Imperial Cancer Research Fund, PO Box 123, Lincoln's Inn Fields, London WC2A 3PX, UK

Received October 19, 1992; Revised and Accepted March 5, 1993

ABSTRACT

A complete set of software tools to aid the physical mapping of a genome has been developed and successfully applied to the genomic mapping of the fission yeast *Schizosaccharomyces pombe*. Two approaches were used for ordering single-copy hybridisation probes: one was based on the simulated annealing algorithm to order all probes, and another on inferring the minimum-spanning subset of the probes using a heuristic filtering procedure. Both algorithms produced almost identical maps, with minor differences in the order of repetitive probes and those having identical hybridisation patterns. A separate algorithm fitted the clones to the established probe order. Approaches for handling experimental noise and repetitive elements are discussed. In addition to these programs and the database management software, tools for visualizing and editing the data are described. The issues of combining the information from different libraries are addressed. Also, ways of handling multiple-copy probes and non-hybridisation data are discussed.

INTRODUCTION

Physical mapping of genomes by constucting ordered libraries of overlapping clones is an essential step in the analysis of the information contained in the genomic DNA sequence, since it gives both a high resolution map of the genome and easy access to the cloned DNA for molecular analysis of genes, transcripts and regulatory sequences.

In this paper we describe a suite of programs for building the physical map of a genome, using data of hybridisations of probes onto unordered clone libraries. These methods were developed to aid the physical mapping of the fission yeast *S.pombe* (1, 10) but are applicable to other hybridisation data.

A very efficient method for generating the data necessary to order a library is to hybridise DNA probes onto filters where the library has been spotted out in a high density grid (2). One can identify the clones positive for a probe as darker signals on an autoradiograph of the filter.

Probes with sequences which only occur once will identify clones lying in a contiguous region of the genome, while multiple-copy probes identify a union of sets of overlapping clones (Fig 1). By applying a sufficiently large number of probes it is possible to order the library into contigs of overlapping clones (3).

The number of hybridisations required to order a library is a function of the insert size of the clones and the depth of coverage of the library. (4, 5). Long clones such as yeast artificial chromosomes—YACs (6) and P1s (7) will require fewer hybridisations than cosmid clones to link them, but the resulting map will be coarser than a cosmid map would be. The frequency of coligation events during the cloning procedure is also important because it is much easier to interpret the results of hybridisations on unfragmented clones.

Because the generation of a clone library is essentially a random sampling of intervals from the genome, even in libraries of high coverage there is a probability that some parts of the genome may not be cloned. Additionally, some regions of the genome may be hard to clone by virtue of their DNA sequences (8).

The major problems faced in contig assembly are the management and visualisation of very large data sets, the efficient selection of probes to minimise the number of experiments and the resolution of contradictions in the experimental data. The creation of a physical map can be viewed as finding the most consistent interpretation of the hybridisation data.

There are advantages using a mapping strategy that integrates information from several clone libraries, (YAC, P1 and cosmid in the case of *S.pombe*). The information obtained from ordering one library may be used in ordering the others, provided that some probes are used on more than one library, thereby providing a set of 'probe-tagged sites' (PTSs) (1). Other sources of information, such as the genetic map of the organism, are also useful in this respect.

MAPPING *S.pombe*

Although the tools described in this paper are quite general, to set the context we give a brief description of the *S.pombe* libraries and mapping strategy. *S.pombe* is a useful test case for prototyping techniques which may be used on larger and more

* To whom correspondence should be addressed

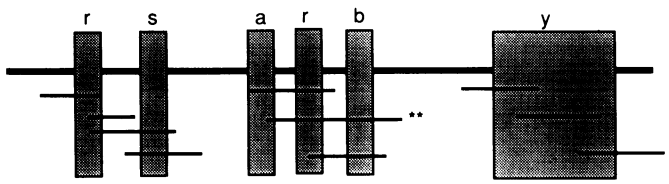

Figure 1. Schematic representation of various hybridisation events. The genome is indicated by the thick horizontal line and the clones as short thin lines below the genome line Probes are the shaded boxes indicating the regions of the genome spanned by the probes. The probes *r, s* are connected by a clone and so would be considered as neighbours. The probe em r has a repeat between the probes *a* and *b*, causing a possible fork in the map. The clone marked with asterisks * spans *a, r, b*, so that in this instance it would be possible to remove the probe *r* without breaking the contig containing *a* and *b*. The probe *y* is a long probe (such as a YAC), which spans many clones, showing that the clones need not overlap with each other.

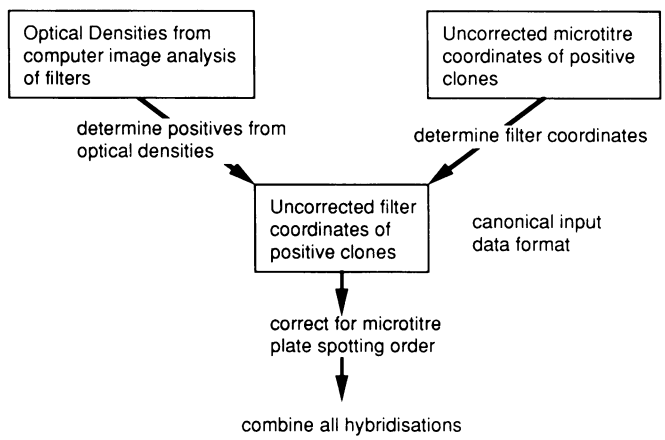

Figure 2. Flow diagram of the database used to hold the hybridisation data. Data from image-analysed or digitised filters are converted into the same canonical input format consisting of a separate file for each hybridisation, each listing the x−y coordinates of the positives on the filter. The files are then combined into a flat file containing all the hybridisations.

complex genomes. The genome of is some 14 Mbp long divided into three chromosomes (9). Probes were hybridised to cosmid, YAC and P1 libraries. The YAC library comprised 1248 clones with an average insert size of 535 kbp, yielding a coverage of 47 genome equivalents. For the P1 library there were 3456 clones with average insert size 70kbp, ie 20 genome equivalents. The cosmid library had an averge insert size of 37 kbp. The total cosmid library contained about 8500 clones, a coverage of 23, but most hybridisations were done on a sublibrary of 3000 clones ie a coverage of 8.5.

The main types of probe used on each library were:

- YAC library: whole cosmids, genetically-mapped markers, YAC-end probes
- cosmid library: whole cosmids, whole YACs, genetically-mapped markers, YAC-end probes, P1 end-probes
- P1 library: whole cosmids, whole P1s, whole YACs, genetically-mapped markers.

The basic strategy was top-down: first order the YAC library by hybridising cosmids and genetic markers (1), and then order the cosmid library (10), using the cosmids hybridised to the YACs as a high density PTS map. The other cosmid probes were picked by sampling without replacement. The P1 library was ordered in parallel and used to bridge any gaps between the cosmid contigs (10). Although not considered further in this paper, several cosmid pools and oligonucleotides were also hybridised to the cosmid library.

SYSTEM, DATABASE AND SOFTWARE TOOLS

System and database

All the software was written in C and runs on a network of SUN SPARCstation I, II and IPX, running SUNOS 4.1.1 and the OpenWindows window manager, version 3. The XView toolkit was used for development of the graphics-based applications running under the X Window System. With the exception of these applications, the programs can be run from Macintosh computers and IBM PC compatibles connected to the network. The programs and data files have distinctive icons, providing a user-friendly graphical interface when run under OpenWindows.

The hybridisation data were held in a flat file data base, with C programs reponsible for data management and collation. Hybridisations of probes onto filters were either digitised or automatically image-analysed. The result of each hybridisation experiment could be input as a list of the filter x−y coordinates of positive clones, or a list of positive microtitre well coordinates or the optical density value of each spot on the filter. These data were held in separate files, which were converted via a series of filter programs into a canonical file format, listing just the clones positive for a given probe. After making corrections for the spotting order of the microtitre plates all the data were combined into two files, one listing the probes used and the other listing the hybridisations of all probes to each clone. These flat files were used as inputs to the contig-building software, together with ancilliary files giving information about the positions of any mapped probes and any contigs that had been determined previously. Fig 2 shows a flow diagram of the database.

The data from cosmid, P1 and YAC libraries were processed in parallel databases which were combined when required by the analysis programs.

Software tools

The programs and a user-manual (11) are available from the authors and the EMBL software library (Netserv@EMBL-Heidelberg.DE) . The majority of the applications access the data through a small library of function calls, so that instead of using the flat files produced by the database they can be interfaced easily to other databases without significant code changes. Those programs that filter or rearrange the data write output files in the same format as the inputs, so that they may be used as inputs to the other applications. There are three types of data-file used by the programs; hybridisation data, (the flat files produced by the database), contig data (lists of contigs, ie ordered probes) and map data (a list of probes ordered from the genetic map or from some other map). Table 1 lists the programs, sorted by function, and input/output.

select is a menu-driven program for choosing subsets of hybridisation data. A user can perform Boolean operations to include or exclude clones which hybridise to particular probes or classes of probes. **unhit** is a non-interactive program for selecting clones and probes, which also gives a list of clones not yet hit by a probe. This was used to pick further cosmid probes

Table 1. A list of the programs, sorted by function.

program	purpose	mode	inputs H	inputs C	inputs M	outputs H	outputs C	outputs M
select	data-selection	interactive	H	.	.	H	.	.
filter	data-selection	non-interactive	H	.	.	.	C	.
unhit	data-selection	non-interactive	H	.	.	H	.	.
show	PostScript display	non-interactive	H	.	.	.	.	.
xvshow	Xview display	interactive	H	.	M	.	.	.
xvedit	Xview contig editor	interactive	H	C	.	H	.	.
reorder	order clones to probes	non-interactive	H	C	.	H	.	.
probeorder	order probes	non-interactive	H	.	M	H	C	.
contigorder	examine contigs	non-interactive	H	C	M	.	C	.
costig	order probes	interactive	H	.	.	.	C	.
barr	order probes	non-interactive	H	.	.	H	C	.
yacorder	order contigs	non-interactive	H	C	M	H	.	.

The input and output files types for each program are coded H—hybridisation data; C—contig data; M—map data.

for *S.pombe* when sampling without replacement. **filter** removes clones which are well contaminants, or which hybridise to unrealistically large numbers of probes and thus are very likely to contain highly repetitive elements or vector concatamers.

Figs 3 and 4 were generated by a flexible PostScript generator, **show**, which can either summarize the entire data-set on a single page, or show it in greater detail spread over several pages. The display treats the data as a matrix, with clones as rows and probes as columns. Where a clone and probe hybridise the corresponding row-column intersection is shaded depending on the level of hybridisation. Annotations or labels may be attached to the clones and probes.

If the data have been ordered into contigs then the positives should occur in overlapping runs (Fig 3), and inconsistencies in the data are immediately apparent by eye because all the hybridisations to each clone are visible, including those which do not fit well with the current order of clones and probes. This is in contrast with the usual representation of a physical map, where clones are summarised as intervals which have been packed into as few lines as possible (Fig 1).

xvshow is an XView analogue to **show**, displaying a scrollable portion of the clone-probe matrix on the screen of a workstation running the OpenWindows window manager.

reorder is a tool to reorder a set of clones to a given order of probes. We describe in the next section methods for building contigs which rely on ordering the probes rather than the clones. A probe order can then be fed into **reorder** to generate the clone order, which is later displayed using **show**. A user can optionally specify a set of 'sleeping probes', which are ignored when ordering the clones but which are output next to the probes used to order the clones. Consequently one may check the consistency of the contigs found with one set of probes (eg cosmid probes on cosmid filters) against the hybridisations of another set (such as YAC probes on cosmid filters).

xvedit is a contig editor running under XView which allows the user to edit probe orderings, moving, deleting and inserting probes and contigs and fitting the clones to the resulting orders. It is essentially a graphical interactive version of **reorder**.

The programs **probeorder, barr** and **costig** for ordering the data are described with their algorithms in the next section.

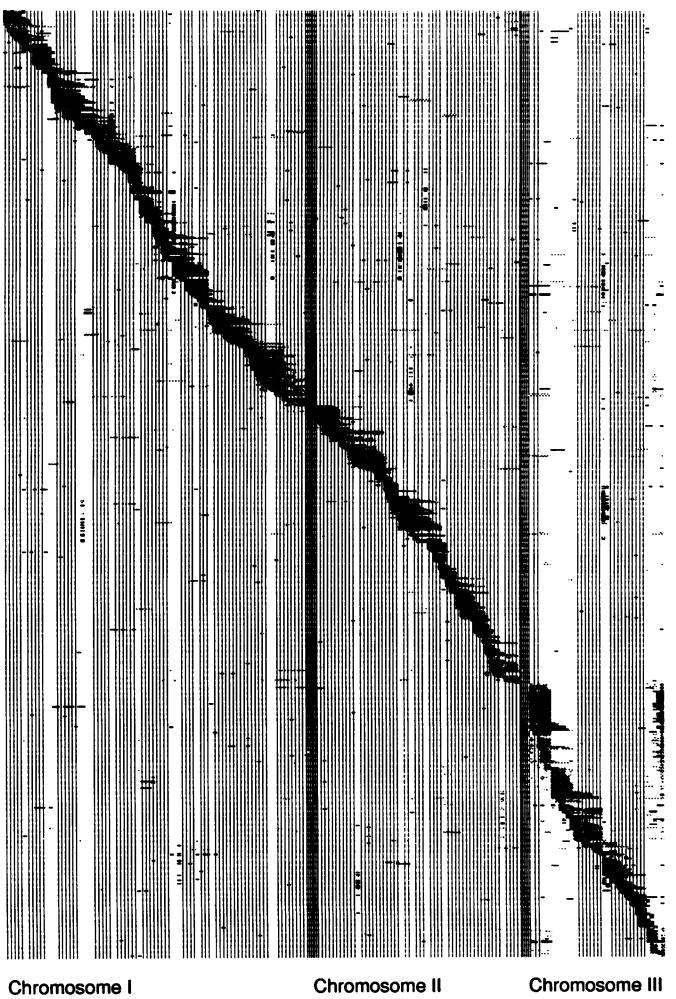

Figure 3. The YAC map for *S.pombe*. The probes correspond to columns and the clones to rows. All the data were used by the simulated annealing algorithm, whilst the filtering algorithm filtered out those probes without a vertical black line (as potentially repetitive), and the grey clones (as potentially coligated). The figure was produced by the **show** program, with some hand-editing of the PostScript to delete the vertical lines of the omitted probes.

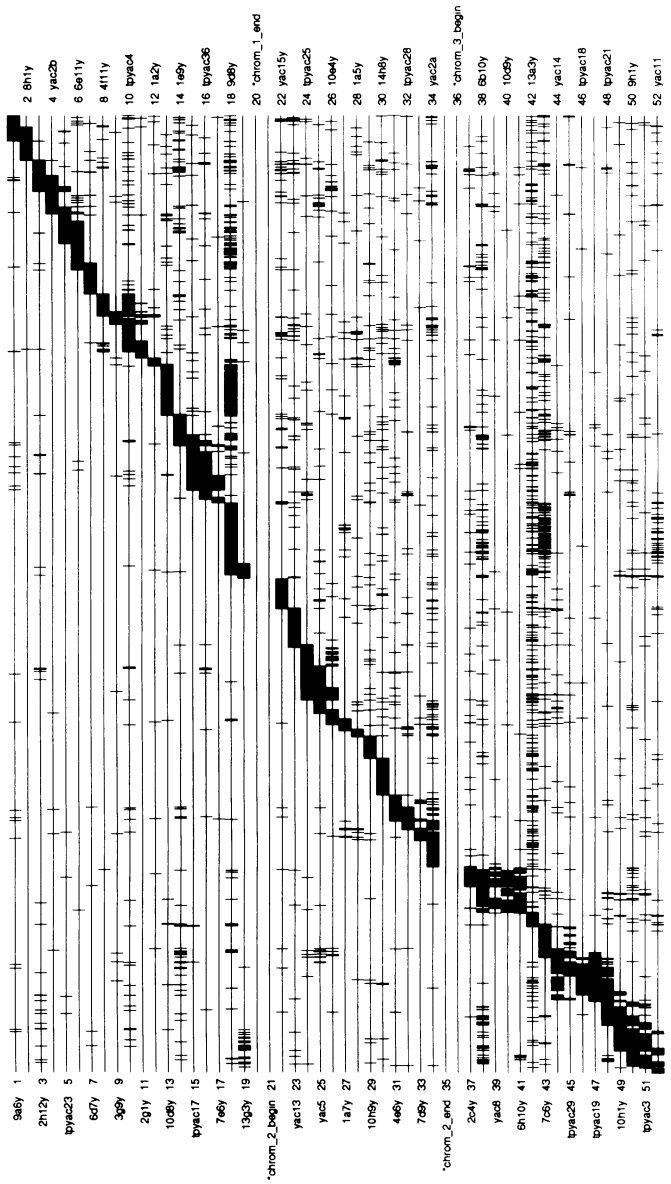

Figure 4. The *S.pombe* map generated by **show**, displaying the fitting cosmid clones to an ordered set of YAC probes. The grey regions indicate either chromosome breaks or missing data. Possibly chimeric YACs (eg 9d8y) are indicated by the clusters of positives way from the main diagonal.

ALGORITHMS FOR ORDERING LIBRARIES

In principle, ordering clones by hybridisation of single-copy probes is trivial, since any pair of clones which share a probe should overlap with the probe. However, in practice, the high level of experimental 'noise' makes such a simple approach untenable because most clones will have false neighbours. Noise in this context is due to various sources:

- random false positives and negatives,
- clones containing coligated inserts
- clones containing internal deletions
- probes containing repetitive elements.

We describe two algorithms which are robust enough to order libraries in the presence of noise. One is based on distance measures and simulated annealing, the other on the application of heuristic rules to clean the data into a consistent set.

Both methods order probes rather than clones. Since the number of probes is much less than the library size, it is more efficient to order the probes first and then fit the clones to the probe order automatically. Also, when comparing two probes we are averaging information over the large number of clones, whereas when comparing two clones we are averaging over the small number of probes, so a probe−probe comparison has a higher information content than a clone−clone comparison.

A consequence of probe ordering is that a contig is defined as a sequence of probes, rather than clones, and a map as a sequence of probe contigs. This makes for a very compact representation of a map; for example the YAC map in Figure 3 can be completely specified by a file listing the 180 probes in order, with the chromosome/contig breaks marked. The probe map can easily be edited manually with a text editor to modifiy the positions of misplaced probes using additional information (e.g. genetic map). All the probe ordering programs produce output files of probe contigs.

Ordering probes using distances

Consider the following simple model: Clones are assumed to be independently and randomly distributed along the genome, and to all have the same length, 1, (in suitable units). Assume further that the probe lengths are sufficiently small that they may be treated as points in comparison with the clones, and that the probes are all single-copy.

Suppose that two probes a,b are a distance $d_{ab} < 1$ apart on the genome. Then the distribution of the number r_{ab} of clones which hybridise to both probes conditional on n_{ab} clones hybridising to either probe is binomial $B(n_{ab}, 1-d_{ab})$, since the clones are independent and the probability that a clone hybridises to both probes given that it hybridises to one is $1-d_{ab}$. Hence the maximum-likelihood estimate of the distance between the probes is

$$\hat{d}_{ab} = \frac{n_{ab} - r_{ab}}{n_{ab}} \qquad (1)$$

The maximum-likelihood estimate is unbiased, with minimum variance, so it is a good estimate to use. The distance $\hat{d}_{ab}$ is 'short-sighted', in the sense that if the probes are more than one clone-length apart then the estimated distance is always unity. If the probes have identical hybridisation patterns then their estimated distance is zero. The distance is formally the same as that used in (12) to estimate the breakage frequency of markers hybridised with radiation hybrids, and as an estimate of the meiotic recombinant frequency in genetic mapping.

The problem of ordering the probes may be cast into the travelling salesman problem; find that circular ordering of the probes with the minimum total path length (13). For a permutation π of the probes, the path length is

$$P(\pi) = \sum_i \hat{d}_{\pi_i \pi_{i+1}} \qquad (2)$$

We used the simulated annealing algorithm described in (14) to minimise the path length of the probe order. Our implementation is derived from the C-routines in (15). It takes an initial order of objects and applies random changes to the order, viz sub-order reversals and sub-order translations (13).

If a change decreases the path length it is always accepted, whilst changes which increase the path length are only accepted with a probability depending in the current annealing temperature and the magnitude of the change. Since each inter-probe distance need be calculated only once the execution time is dependent primarily on the number of probes, not the number of clones.

The output probe order of the annealing is then broken into a set of probe contigs, with either no or very few clones connecting the last probe of one contig to the first probe of the next. An adjustable cutoff distance value is used to determine where the contig breaks occur. Any probes which have been previously mapped provide a means for ordering and orienting these contigs into their correct positions on the genome, for if two contigs contain neighbouring mapped probes then it is likely the contigs should be adjacent, even if there are no hybridisations linking them. To order the cosmid library of *S.pombe*, the PTS map established by previously ordering the YAC library was used in this way.

Once the order of probes and contigs is established the clones are fitted to the probe order, using the algorithm described in Section 4.3. All potentially inconsistent hybridisations, ie any hybridisation linking a clone in one contig to a probe in another are listed. If, after being ordered using a map, a pair of linked contigs are adjacent then these links are more likely to be genuine. If a clone has also been used as a probe then the program also checks if the probe and clone are assigned to the same contig.

The algorithm has been implemented in a program called **probeorder**. It can be used to order any set of single-copy probes which are all approximately the same size, such as cosmid and marker probes on YAC filters, YAC probes on cosmid filters or cosmid probes on cosmid filters. In the latter two cases the probe distances should be interpreted just as dissimilarities since the probes cannot be considered as points in comparison with the clones. However the results obtained from ordering cosmid probes on cosmid filters indicate that the assumption that the probes are points is not critical. In the case of ordering YAC probes, a modified distance measure was used in which the influence of each cosmid clone c was weighted in proportion to $1/n_c$, where n_c is the number of YAC probes positive for that cosmid clone. This was to downweight the effects of a subset of cosmid clones containing a copy of the rDNA repeat, and which were positive for the majority of YAC probes. The ordering of YAC probes was harder than for other probes types in that the highly variable length of the YACs meant that some YACs were contained entirely within others, and some YACs were chimeric, requiring some manual adjustment of the order.

The algorithm takes about 50 CPU seconds on a SUN SPARCStation II to order 180 probes and 1150 clones from the *S.pombe* YAC library, and 247 CPU seconds to order 667 probes and 2837 clones from the *S.pombe* cosmid library, including the phase of contig ordering and consistency checking.

If the user only wishes to check the consistency of a predetermined set of contigs then a program **contigorder** may be used instead. This does not order the probes but instead takes a contig list as input and computes every inconsistent hybridisation (defined as any link between a probe in one contig and a clone in another).

Ordering probes using heuristics

The major obstacle in contig ordering is experimental noise. In the case of 'ideal' noise-free experimental data, various simple algorithms, exploiting graph structures or tree-search techniques, can successfully order the library. A general outline for any such algorithm will look as follows:

- For each probe find all neighbouring probes, ie probes linked by jointly positive clones.
- Order all probes relative to their neighbours according to the procedure given in pseudocode in Fig. 5.

```
while an unordered probe exists
        start at some random unordered probe X
elongation:
        mark current probe as ordered
        find its least/most distant neighbour Y in one direction
        if no unordered neighbours exist
                if only one direction is searched through
                        take X as current probe again and change direction
                        goto elongation
                else continue (next iteration of while loop)
        if the most distant neighbour Y is found
                mark all probes common for both neighbourhoods as ordered
                (being between these two)
        take this neighbour Y as current probe
        goto elongation
```

Figure 5. An algorithm for ordering ideal hybridisation data.

In the neighbourhood of a given probe, X, the *most distant neighbour* can be defined either as that probe whose own neighbourhood shares the smallest number of probes with X, or/and as that probe with the smallest number of clones connecting it with X. One can define the *least distant neighbour* of the probe X analogously.

The *gotos* in the pseudocode can be replaced by a more elegant recursion. This algorithm will produce a relative order of probes (ie a path in a graph or two deepest branches in a tree) for each group of clones which could be relatively ordered, i. e. for each contig. The choice between searching for the least or most distant neighbours may depend on the experimental (or even presentational) needs because in the former case the algorithm finds more detailed and possibly informationally redundant order of probes, while in the latter case it finds a minimal set of probes connected by clones spanning large regions of the genome. Combining both cases may be also useful for checking the consistency of both orders by superimposition and comparison of them.

However, it is easily seen that a single false positive will create a fork in a map (a loop in a graph or a shared branch in a tree). Realistic experimental noise together with the 'natural' forks caused by repetitive sequences will result in unpredictable and far from real maps. So a stage of initial filtering of the data becomes a prerequisite.

If a fork results from repeats in the genome (and is therefore likely to violate the neighbourhood rules given below), it is reasonable simply to make a break in the contig because it represents a limit of the experimental technique used and other approaches using longer probes or clones may help to close the gap. In many cases, however, neglecting the data from a probe containing a repeat allows one to find a correct connection to elongate the contig (Fig. 1).

In contrast, random false hybridization signals and non-contiguous clones yield additional false neighbours which can

be identified by lower numbers of links with a given probe. Neglecting of clones producing these links is the simpliest way of resolving the corresponding forks, although a more careful analysis of hybridization data for such probe pairs and clones linking them could help to reduce the level of noise.

With data filtering one pays the price of decreasing the effective library redundancy and increasing the number of hybridisations. In so far as the goal is to extract a 'clean' data set of ordered clones and probes, but which is still large enough to minimize the number of contigs, a high initial redundancy implies that less additional effort is needed to close any remaining gaps when all available probes are used and all contiguous clones are ordered. The mapping of *S.pombe* in YAC clones clearly illustrates this principle. For a detailed theoretical discussion on how the requirement of multiple links in contig construction affects the experimental effort for libraries of a given clone redundancy see (4).

We combined a method for finding probes and clones which may cause map forking with the ordering algorithm of Fig. 5. A few simple heuristic rules are used to identify 'suspect' clones and probes and to find each probe's neighbours simultaneously. Then the 'suspects' are presented to the user who decides upon removing them from the analysis. After his decision, contigs are built according to Fig 5. The rules for filtering are as follows:

- Considering the clones hit by at most N probes, the number of neighbours for any probe must not exceed $2(N - 1)$.
- For two probes to be neighbours, the number of clones n positive for both of them must be > 1. For the *S.pombe* YAC library the redundancy allowed us to use $n = 3$.

Thus the process of ordering a library consists of several iterative stages of filtering out clones which connected pairs of probes less then n times. At each stage only clones hit less then N times are analysed. If, after filtering the clones, a probe having more than $2(N - 1)$ neighbours is found, it is reported as a 'suspect' one and the user may remove it from the analysis and repeat the procedure. During ordering probes, a constraint on the number of neighbours in any one direction (which cannot exceed $N - 1$) is checked.

The algorithm has been implemented in two versions (using least and most distant neighbours). The least distant neighbour version (**costig**) was more applicable to ordering the cosmid library under the scheme of sampling without replacement and has a menu-driven interface allowing, among other options, the output of any single contig specified by a probe belonging to it. The most distant neighbour version (**barr**) was used for ordering the YAC library. It always produced the minimum set of ordered probes. The algorithm is fast, taking under a second to order 1150 clones and 180 probes from the *S.pombe* YAC library and under 3 seconds to order 667 probes and 2837 clones from the *S.pombe* cosmid library on SUN SPARCStation II.

Fitting clones to a probe order

Once the correct order of probes has been established it is easy to fit the clones to this probe order, using an algorithm which essentially places each clone on that section of the probe order where it has the highest density of positives. An ordering of N probes imposes a natural integer-valued coordinate system on the genome, in which each probe occupies one of the positions $i = 1,2,3...N$. It is then sufficient to determine each clone's start and end coordinates, say (s,e). Define a match $a > 0$ and a mismatch $b < 0$ cost to score the fit $f_{se}(c)$ of the clone c to the interval (s,e),

$$f_{se}(c) = \begin{cases} 0 & \text{if } e < s \\ \max(0, f_{se-1} + ah(c,p_e) + b(1 - h(c,p_e))) & (N \geq e \geq s \geq 1) \end{cases}$$

where p_i is the probe occuring at position i and $h(c,p)$ is an indicator variable indicating whether the clone c and the probe p hybridise. The values of (s,e) at which $\hat{f}(c) = f_{se}(c)$ is a maximum define the best-fitting range for the clone c.

For fixed s, the $f_{se}(c)$ values can be constructed iteratively starting from $e = s$. Once $f_{se}(c)$ equals zero then there is no need to consider larger values of e. Figs 3 and 4 were generated by this algorithm, with parameter settings $a = 2$, $b = -3$. The method is fast, taking under a second to fit 1150 clones to 180 probes on a SUN SPARCStation II. This is the algorithm used by all programs that fit clones to the predetermined order of probes.

EXAMPLE: ORDERING THE YAC MAP OF *S.pombe*

The contigs of the cosmid and YAC libraries found by the two methods were essentially identical except for minor differences in order neighbouring probes having very similar or identical hybridisation patterns and which could be easily swapped. That was a convincing check for consistency of the resulting map. The map was also verified experimentally by digesting a spanning subset of 40 YACs with NotI (10), and comparing fragment digests with inferred order of YACS from the hybridisation data.

Because simulated annealing is a stochastic algorithm, different runs of **probeorder** will not necessarily produce the same output probe order. However, we found in practice that in different runs very similar contigs were generated, the differences between runs being confined to contig breaks and to regions in the probe order where the probes were repetitive. If there was no map information available then the order and orientation of the contigs was random, but the order of probes within each contig was stable.

In 10 runs on the complete YAC data-set, using a distance cutoff of $d = 0.85$ to determine contig breaks, the 'correct' probe order for both chromosomes I and II was found on two occasions, each time with the same path length of 7381. The *rDNA* repeat at the end of chromosome III was always placed incorrectly next to the *rDNA* repeat at the beginning. On the other runs the final path length was slightly higher, with the maximum value over all runs of 7392, and with errors in the probe orders being confined to the centromeres of one or both of chromosomes I and II (e.g., one half of chromosome 2 would be joined to chromosome III). The remainder of the probe order (between the each telomere and the corresponding centromere) was correct in all runs. It is straightforward to edit the errors manually.

In this dataset the lowest path length found did correspond to the correct order for chromosomes I and II. However, the fact that the other local minima found by the algorithm were all less than 0.15% greater than this value indicates that the path length would not necessarily have a minimum coinciding with the best probe order in other data sets containing many repeats, a fact illustrated by chromosome III. Consequently it is always worth considering other local minima found by the annealing process, as these may correspond to better alternative probe orderings. Also, other measures of pairwise probe distance/similarity may prove to be better-suited to future applications than the maximum-likelihood distance used here.

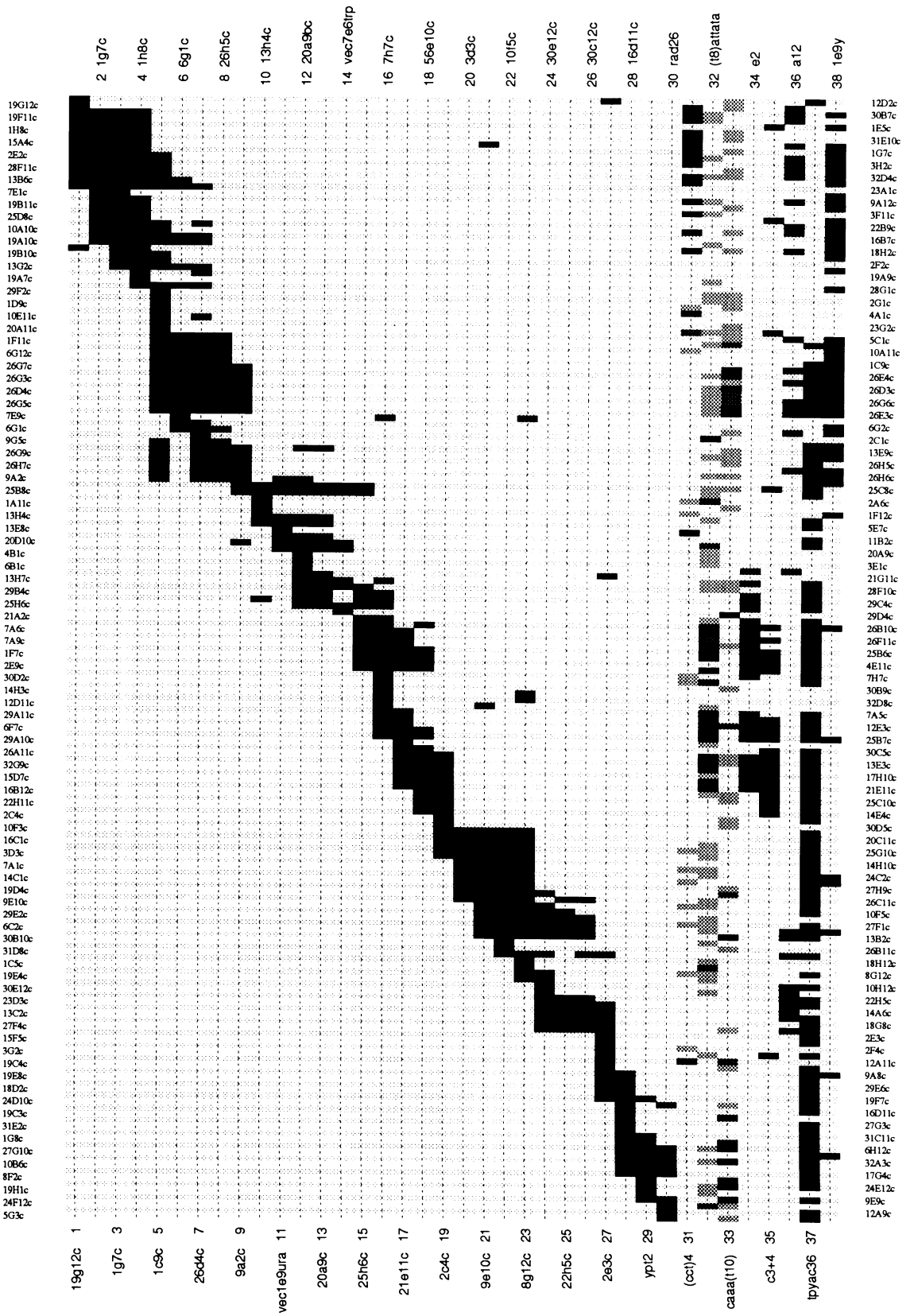

Figure 6. An example output of **show**, displaying one of the cosmig contigs constructed in the middle of the *S.pombe* project. Cosmid clones in this contig were fitted to the order of cosmid probes (vertical lines 1–30) by the program **reorder**, while 8 other probes were specified as 'sleeping' ones (31–33—oligonucleotide probes, 34–36—cosmid pools, 37–38—YAC probes). Breaks in 'sleeping' probe coverage, most clear for YAC probes, correspond to the groups of clones probably belonging to other regions of the genome and are usually hit by only one cosmid probe. As **reorder** fits all the library clones into a given probe order, false positives become apparent when comparing results of hybridisations of different types of probes. Image analysis artifacts result in a higher noise rate for the oligonucleotide probes.

The order of YAC probes shown in Fig 4 was also produced by **probeorder**, except that the order of the contigs was rearranged into map order, and the chimeric probes 9d8y and 10d8y (which do not really have a unique position on the genome) have been moved. The Figure omits the clones and probes spanning the *rDNA* region, because the majority of these clones also hybridise to over half of of the other YAC probes, and so it is not possible to place these clones and probes at a single position on the map.

To demonstrate the robustness of the heuristic filtering procedure, the map of *S.pombe* was constructed using **barr** on the complete YAC data set. During the iterative runs of the program when the parameter *N*—the number of hits per clone (see section 4.2) was changed from 2 to 7 and all the 'suspect' probes reported by the program were deleted together, not taking into account the information about repetitive probes. Thus human intervention was excluded in this 'blind' approach, to demonstrate possible problems in cases where the there was no genetic or PTS map.

Fig. 3 shows the result superimposed onto the final YAC map of *S.pombe*. Clones deleted in the analysis are indicated as grey horizontal lines, with the non-deleted clones shown black. Similarly, the black vertical lines correspond to the ordered subset of probes, while the lines are omitted for the probes which were filtered out, resulting in white space breaks.

It is easily seen that all the probes producing 'blocks' of extra positives outside the main diagonal (like those hitting the centromeric regions of all three chromosomes) are successfully filtered out and most (about 80%) of the coligated clones or those containing repeats are excluded from the analysis. The remaining subset of the initial 'raw' data allows the program to reconstruct the complete maps of the chromosomes I and II as well as most of the chromosome III map.

The wide gap in the chromosome III (accounting for 1Mbp of the *rDNA* repeat) is in fact not a gap but an undetected overlap of two groups of clones having 5 deleted probes in common. This is an artifact of this 'blind' filtering approach, when information about genetically mapped probes in this region was intentionally ignored and all 'suspect' probes were deleted together. Obviously, if probes in this region were deleted one by one, starting from the probes known as repetitive (containing ribosomal DNA or belonging to telomeres in this case), less of them would be deleted and the overlap would be successfully detected.

This example also demonstrates the basic principle of handling the repetitive elements: probes in the region containing this element are deleted one by one until two non-repetitive probes on both sides of the region are found which are connected by a minimum number of clones, linking these probes into a contig. Thus all the probes producing vertical 'blocks' of positives outside the main diagonal in Fig. 3 are removed from the analysis and the resulting contigs can be found successfully. In the case when a clone length is less than that of a repetitive element (or a total length of a group of adjacent repeats), a break in the contig is inevitable and can only be closed by using longer clones which can bridge the repeat. Therefore for highly repetitive genomes it is likely that only maps containing clones from libraries of different types, such as cosmids and YACs, can be constructed and an optimal strategy of mapping of different resolutions, based on previously established probe-tagged sites, must be elaborated.

The high repeat frequency of the chromosome III and close resemblance between its telomeric sequences (effectively closing it into a circle), plus the higher number of clones hybridising to probes mapped to other two chromosomes, demonstrate the possible problems with all types of algorithm with more complex genomes.

COMBINING INFORMATION FROM YAC AND COSMID DATA

The hybridisations of YACs on cosmids together with the order of the cosmid probes hybridised to the YAC filters were used to help order the cosmid contigs and check their internal consistency. The basic idea is that the cosmid probes that make up a contig should hit cosmid clones which are all hit by one or two YACs. Suppose M probes have been ordered on the YAC map and the probe p_m is positioned at coordinate m on the map ($m = 1,2,...M$). For a YAC probe y, and any other probe p, that have both been hybridised to the cosmid library, define the association $\theta(y,p)$ of y and p as the proportion of cosmid clones positive for probe p which are also positive for probe y. Then the association of the YAC y to position m is defined as $\theta(y,p_m)$. The region on the map spanned by the YAC should correspond to high θ values. If the YAC y is chimeric then it will have more than one such region. For a probe q which is not in the YAC map, the association of q to position m is defined as

$$\theta(q,m) = \sum_{y} \theta(y,q)\ \theta(y,p_m) \qquad (3)$$

The association of the cosmid contig c, containing the set of probes S_c, with position m is defined as

$$\theta(c,m) = \sum_{q \in S_c} \theta(q,p_m) \qquad (4)$$

This method effectively computes a profile for each probe and contig, indicating the most probable region of the contig on the map. The profile may be thought of as a set of fake hybridisations with the set of M mapped probes and the probe/contig, and consequently may be displayed using **show**. The algorithm is implemented in the program **yacorder** and is used for positioning contigs which do not contain any mapped probes and so cannot be positioned directly.

DISCUSSION

We have learned from our experiences with *S.pombe* the importance of combining information from different clone libraries, and from genetic or PTS maps. Relating the libraries by hybridising clones from one library onto another has a twofold advantage—the maps can be unified and probes can be chosen to close gaps as economically as possible. Whilst computational methods are an essential tool in building maps, if only because of the sheer volume of data, at present human intervention is still needed to resolve inconsistencies. Using a variety of ordering algorithms which complement one another provides a check that the contigs they produce are not just artifacts.

The ability to display hybridisation data from either the whole dataset or any subset (eg a contig or group of contigs) using **show** was very useful, particularly in conjunction with **reorder**. Examples of its use include checking if gaps in the cosmid map are actually covered by YAC or P1 probes, checking for consistent ordering of different types of probes and even to interpret the hybridisation results of multiple-copy probes. It was

also used to aid selecting cosmid clones which were to be used as probes on the cosmid filters. The cosmids were picked in batches of about 50 from the set of unhit clones (ie by sampling without replacement), and so to ensure that the cosmids in each batch were from different regions on the genome (and hence to maximize efficiency) the unhit clones were reordered against the ordered YAC probes. Cosmids were then picked from different locations of the resulting map.

Although the ordering algorithms were developed for hybridisation mapping projects, the software can be used with other types of data which can be interpreted as hybridisation-like events. For instance, restriction fragments of the clones in the library can play the role of probes, and the presence of a fragment in a clone digest will be equivalent to a positive hybridisation between the corresponding clone and 'probe'. Due to errors in determination of fragment lengths and similarities between lengths of fragments in different parts of a genome, this interpretation will be closer to the hybridisation with multiple-copy probes (plus additional 'noise' from fragments containing the vector component of a clone). However, using the information about the clone order from hybridisation data it will be much easier to infer a restriction map of a genome (and verify a hybridisation map in parallel) from these results than to construct the complete two single-digest plus one double-digest map from individual restriction maps of each clone. A program performing this task is currently under development.

We investigated integrating information about mapped probes directly in the probe ordering. Given a probe order π, and a subset mapped with suborder π_m, then one can compute a similarity measure $M\pi,\pi_m)$ between the orders. The function to be minimised by the annealing then becomes $P(\pi) - M(\pi,\pi_m)$. Suitable candidates for M are a dynamic-programming-type alignment score (16) of π,π_m or a simpler and faster scoring scheme based on the numbers of pairs of mapped probes which are adjacent in π_m and are separated only by unmapped probes in π.

However, this approach was not as successful as that implemented in **probeorder**, in that where the map and hybridisation data are in in serious conflict the contigs generated will be a poor compromise between the two. It is better to generate the contigs according to the hybridisation data and then automatically check them for consistency with the map. Then the cause of an inconsistency, (perhaps a repetitive probe a mislabelled autoradiograph), can be tracked down.

We cannot as yet simultaneously order probes of markedly different sizes, such as cosmids and YACs (although clones of different sizes are acceptable). One solution to this problem would be to assign a length to a probe, rather than treating it as a point, so that short probes could be contained within longer ones. One can define, for some probe order, the consistency measure $\Sigma_c \hat{f}(c)$ for the total fit of all the clones to the probes, where $\hat{f}(c)$ is the fit of the clone c, either as defined above or extended to allow for probes having a length or being repetitive. A potential criterion for ordering the data would be to choose that order which maximises the consistency, and the simulated annealing algorithm described earlier could be adapted to do this.

Although unnecessary with *S.pombe*, in cases where there is uncertainty whether probe a is positive for clone c, the hybridisation event may be given a grey-level $0 < h_{ac} < 1$, with 1 indicating a certain positive and 0 a certain negative. The inter-probe distance measure can easily be extended to use this information by redefining $r_{ab} = \Sigma_c(h_{ac}h_{bc})$ and $n_{ab} = \Sigma_c(h_{ac} + h_{bc}) - r_{ab}$. Similarly, when ordering using heuristics, the number of links between two probes can be weighted according to the grey-levels

In the *S.pombe* cosmid library, some probes were only hybridised to a subset of 1500 clones, resulting in missing data. When comparing two probes, one of which has been hybridised to the complete library and the other to a subset, the distance is calculated just using the clones in the subset, as the other clones provide no information about this particular comparison. This is a consistent way of treating missing values, in that the expected value of the estimated distance is independent of the size of the library, although the standard error is larger with smaller libraries. When counting the number of links between two probes, missing clones are clearly of no relevance. Of course, pairs of probes that have been hybridised to mutually disjoint libraries cannot cannot be compared in this way. In this situation the distance/number of links between the pair could be calculated from the 'shortest path' connecting them through a sequence of intermediate linked probes.

The analysis software described in this paper does not order multiple-copy probes. Although the use of single copy probes is efficient for small genomes like *S.pombe*, theoretically hybridisation based fingerprinting techniques (3, 17) using multi-locus probes are more attractive for larger genomes. They generate data at a higher rate (ie yield more positives per hybridisation) and are less sensitive to repetitive sequences (2).

However, hybridisation based fingerprinting requires a high reliability of data acquisition (e.g. during image analysis) and pose considerably more difficult problems for map ordering algorithms. In contrast to single copy probes, the number of loci and their positions for multi-locus probes are usually unknown. Algorithms based on ordering clones are an obvious avenue to persue here, provided the clones can be treated as intervals rather than points. Our current algorithms are based on calculating the likelihood ratio of each pairwise clone overlap, combined with the use of the minimum spanning tree and simulated annealing algorithms to order the clones.

Another approach, based on an extension of the most/least distant neighbour algorithm may be as follows. Not the individual probes but the longest unique 'probe patterns', containing several adjacent probes must be considered as neighbourhood elements for all such probe patterns. After establishing and connecting the neighbourhoods, all individual probes in a pattern can be ordered easily. However, a stage of filtering, including a check for the integrity of a probe pattern in the presence of experimental noise requires a more complicated filtering procedure than that used for single-copy probes. In contrast to filtering the single-copy hybridisation data, false negatives have a higher effect on the correct determination of patterns.

All the programs described in this paper are available from the authors or from the EMBL software server Netserv@EMBL-Heidelberg.DE in the compressed tar-file icrf_contig.tar.Z.

REFERENCES

1. Maier, E., Hoheisel, J. D., McCarthy, L., Mott, R. F., Grigoriev, A. V., Monaco, A. P., Larin, Z. and Lehrach, H. (1992) Nature Genetics 1, 273–277.
2. Lehrach, H., Drmanac, R. Hoheisel, J., Larin, Z., Lennon, G., Monaco, A.P., Nizetic, D., Zehetner, G. and Poustka, A. (1990) In Davies, K. E. and Tighman, S. (eds), Genome Analysis: Genetic and Physical Mapping. Cold Spring Harbor Laboratory Press, Cold Spring Harbor. pp 39–81.
3. Craig, H. D., Nizetic, D., Hoheisel, J. D., Zehetner, G., Lehrach, H. (1990) Nucl. Acids Res. 18, 2653–2660.

4. Arratia, R, Lander, E. S., Tavare, S. and Waterman, M. S. (1991) Genomics 11 806–827.
5. Ewens, W. J., Bell, C. J., Donelly, P. J., Dunn, P. Matallana, E. and Ecker, J. R. (1991) Genomics 11 799–805.
6. Burke, D. T., Carle, G. F. and Olson, M. V.(1987) Science 236 806–812.
7. Sternberg, N. L. (1990) Proc. Natl. Acad. Sci. U.S.A. 87, 177–181.
8. Clarke, L. and Carbon, J. (1976) Cell 9, 91–99.
9. Fan, J.-B., Chikashige, Y., Smith, C. L., Niwa, O., Yanagida, M., and Cantor, C. R. (1988) Nucl. Acids Res. 17, 2801–2818.
10. Hoheisel J. D., Maier E., Mott R.F., McCarthy L., Grigoriev A.V., Schalkwyk L.C., Nizetic D., Francis F. and Lehrach H. (1993) Cell. In press.
11. Mott, R. F. and Grigoriev, A. V. (1992) Programs for analysing Hybridsation Data—user's manual (unpublished—available from EMBL software server Netserv@EMBL-Heidelberg.DE).
12. Cox, D. R., Burmeister, E., Price, R., Kim, S., and Myers, R. M. (1990) Science, 250, 245–250.
13. Lin, S. (1965) Bell System Tech. J. 44, 2245–2269.
14. Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller A., and Teller, E. (1953) J. Chem. Phys. 44, 1087–1092.
15. Press, W. H., Flannery, B. P., Teukolsky, S. A. and Vetterling, W. T. (1988) Numerical Recipes in C (Cambridge University Press, Cambridge), pp 333–351.
16. Needleman, S. D. and Wunsch C. (1970), J. Mol. Biol. 48, 444–453.
17. Poustka, A., Pohl, T., Barlow, D.P., Zehetner, G., Craig, A., Michiels, F., Euch, E., Frishauf A.M and Lehrach, H. (1986), Cold Spring Harbor Symposium on Quantitative Biology 51, 131–139.