Vol. 13 no. 5 1997
Pages 523-535

FPC: a system for building contigs from restriction fingerprinted clones

C. Soderlund[1], I. Longden and R. Mott

The Sanger Centre, Wellcome Trust Genome Campus, Hinxton, Cambridge CB10 1SA, UK

Received on February 25, 1997 revised and accepted on April 24, 1997

Abstract

Motivation: *To meet the demands of large-scale sequencing, thousands of clones must be fingerprinted and assembled into contigs. To determine the order of clones, a typical experiment is to digest the clones with one or more restriction enzymes and measure the resulting fragments. The probability of two clones overlapping is based on the similarity of their fragments. A contig contains two or more overlapping clones and a minimal tiling path of clones is selected to be sequenced. Interactive software with algorithmic support is necessary to assemble the clones into contigs quickly.*

Results: *FPC (fingerprinted contigs) is an interactive program for building contigs from restriction fingerprinted clones. FPC uses an algorithm to cluster clones into contigs based on their probability of coincidence score. For each contig, it builds a consensus band (CB) map which is similar to a restriction map, but it does not try to resolve all the errors. The CB map is used to assign coordinates to the clones based on their alignment to the map and to provide a detailed visualization of the clone overlap. FPC has editing facilities for the user to refine the coordinates and to remove poorly fingerprinted clones. Functions are available for updating an FPC database with new clones. Contigs can easily be merged, split or deleted. Markers can be added to clones and are displayed with the appropriate contig. Sequence-ready clones can be selected and their sequencing status displayed. As such, FPC is an integrated program for the assembly of sequence-ready clones for large-scale sequencing projects.*

Availability: *The software and manual are available via anonymous ftp to ftp.sanger.ac.uk, directory pub/fpc.*

Contact: *E-mail:cari@sanger.ac.uk*

Introduction

FPC (fingerprinted contigs) is an interactive program for creating contigs from fingerprinted clones. This system is the replacement for CONTIG9 (Sulston *et al.*, 1988) which was used for *Caenorhabditis elegans*. The success of this project (Coulson *et al.*, 1986), has led us to scale up this technique for fingerprinting the human genome (Gregory *et al.*, 1996), to which end FPC provides advanced features for assembling, viewing and editing contigs. In addition, FPC supports marker data to help confirm overlaps and order contigs.

Fingerprinting is a clone comparison technique based on the matching of characteristic fragment sets between clones. The fingerprinting method used at the Sanger Centre generates fragment sets for clones by digesting them with two restriction enzymes and labelling (radioactively or fluorescently) the sticky end of only one of the restriction enzyme sites. The fragments are separated by electrophoresis on thin polyacrylamide gels and the migration distance of the labelled fragments is measured. The resulting fragments are generally called bands and the distance values are inversely related to the actual length. The bands represent a subset of the DNA contained within the clone. For example, in the *C.elegans* project (Coulson *et al.*, 1986), cosmid clones are digested using a *Hin*dIII/*Sau*3A reaction where only the *Hin*dIII sites are labelled, which produces on average 23 bands per clone with distance values that generally range between [200, 3500]. Using this method, two bands are considered the same if their values are within a constant tolerance.

An alternative fingerprinting method uses a complete restriction digest run on an agarose gel and the sizes of the fragments are measured. The fragment set represents the whole clone, and can be used both for building restriction maps and for sizing the clone. For example, the Los Alamos Chromosome 16 project (Stallings *et al.*, 1992) digests cosmid clones with *Eco*RI, resulting in fragment sizes in the range [600, 14000] (Soderlund *et al.*, 1993). Using this method, two fragments are considered the same if the difference in their lengths is within a variable tolerance proportional to the fragment length.

In both methods, a clone fingerprint is defined as a set of measured fragment values and the similarity in the measurements between two clones gives evidence about their overlap. They both may have the following errors and uncertainty: (i) false-positive and false-negative bands; (ii) two bands may have the same measurement but be different bands; (iii) two bands may have a difference that exceeds the

[1]*To whom correspondence should be addressed*

tolerance and yet be the same band; (iv) partial bands. Hence, FPC can be used with either type of data by only changing how the bands are compared. FPC cannot be used for data from multiple complete digests (Gillett *et al.*, 1996), partial digests (Kohara *et al.*, 1987) or double-digest data where two single digests and one double digest are used to build restriction maps (e.g. Waterman and Griggs, 1986).

In contig assembly systems, contigs of overlapping clones are represented in an artificial coordinate system based on the clones' band data. Each clone in a contig has a left and right coordinate, and the overlap between two clones corresponds to the number of bands they share. To calculate a coarse set of coordinates, the probable locations of the clones can be obtained by ordering the clones based on the estimated overlap between each pair of clones (Branscomb *et al.*, 1990; Stallings *et al.*, 1992). Refined coordinates can by obtained by calculating a partially ordered restriction map and aligning the clones to the map (Olson *et al.*, 1986; Soderlund and Burks 1994; Gillett *et al.*, 1995); this requires either extremely high-resolution fingerprints or a high degree of human intervention.

FPC builds a consensus bands (CB) map which can be considered as a low-resolution restriction map as it does not try to resolve all bands. It is used for accurate positioning of the clones and detection of cloning anomalies. Given that CB maps are built using a fast approximation algorithm and inexact data, error accumulates when trying to build a map for a large contig. Therefore, interactive graphics are provided for the user to edit and merge maps. There is no limit to the number of clones it can process. The FPC display of a contig, which can include marker data (see Figure 1), has been invaluable in our preparation of sequence-ready clones.

FPC was designed to support fingerprinting at the Sanger Centre. It has recently been tuned for complete digest data to support the mapping effort at Washington University Genome Sequencing Center (GSC) in St Louis. Various other laboratories have also been using it since the summer of 1996. This paper primarily discusses the algorithm used in FPC V2.6. It also briefly describes the salient interactive graphic features.

System and methods

FPC is implemented on a Digital Alpha 3000 workstation running OSF V3.2. All software is written in C and compiled with the standard Digital C compiler. The software does not use any system-dependent functions, and we expect that it will be portable to any machine with a standard C compiler and sufficient physical or virtual memory. Timing results using 32 and 64 megabases of main memory are given in Table 1. The software has also been tested on a Sun Sparc Station and Silicon Graphics SGI. The manual, executables for various machines, demo files and the source code are available via anonymous ftp from directory://ftp.sanger.ac.uk/pub/fpc.

Table 1. The CB algorithm was timed on a Alpha 3000 with 64 megabytes of main memory and a Sun Sparc 1+ with 32 megabytes of main memory. A '–' indicates that it ran out of memory. The time is in minutes:seconds and is measured using the Unix *getrusage* system call. The Contig column shows the number of contigs, the number of clones in the largest contig. and the number of singletons (i.e. clones not placed in a contig). The Length column shows the total length of contigs in bands and the average length of clone in bands (since the Coulson *et al.* method does not give the actual lengths of bands, the notional length of a clone or map is one unit per band). The two datasets with 1799 clones are the same except that the first run used a much less stringent cut-off than the second. Note that the low stringency causes many pairs of clones to appear to overlap and an internal structure is made for each, causing it run out of memory on the Sun

Clones	Contigs			Length		Alpha 64meg		Sun 32meg	
	Number	Max	Singletons	Total	Clone	User	System	User	System
36	1	36	0	117	21	0:02	0.00	0:14	0.00
300	38	29	115	1112	21	0:18	0:42	0:53	0:07
365	38	50	151	1983	35	0:37	1:25	2:11	0:12
1695	173	128	855	7466	32	5:13	15:54	16:56	3:53
1799	29	1489	68	1027	19	2:20	5:13	–	–
1799	94	348	212	2843	19	3:15	6:52	12:19	4:22
2250	193	410	718	7916	27	6:50	18:35	22:12	6:26
3980	655	164	1264	11 553	12	7:23	13:24	36:28	22:35
6068	619	369	2085	22 705	25	32:45	84:54	112:55	41:58
11 608	1531	393	2702	29 365	13	65:80	75:53	–	–
17 738	868	928	6081	22 545	16	158:34	222:44	–	–

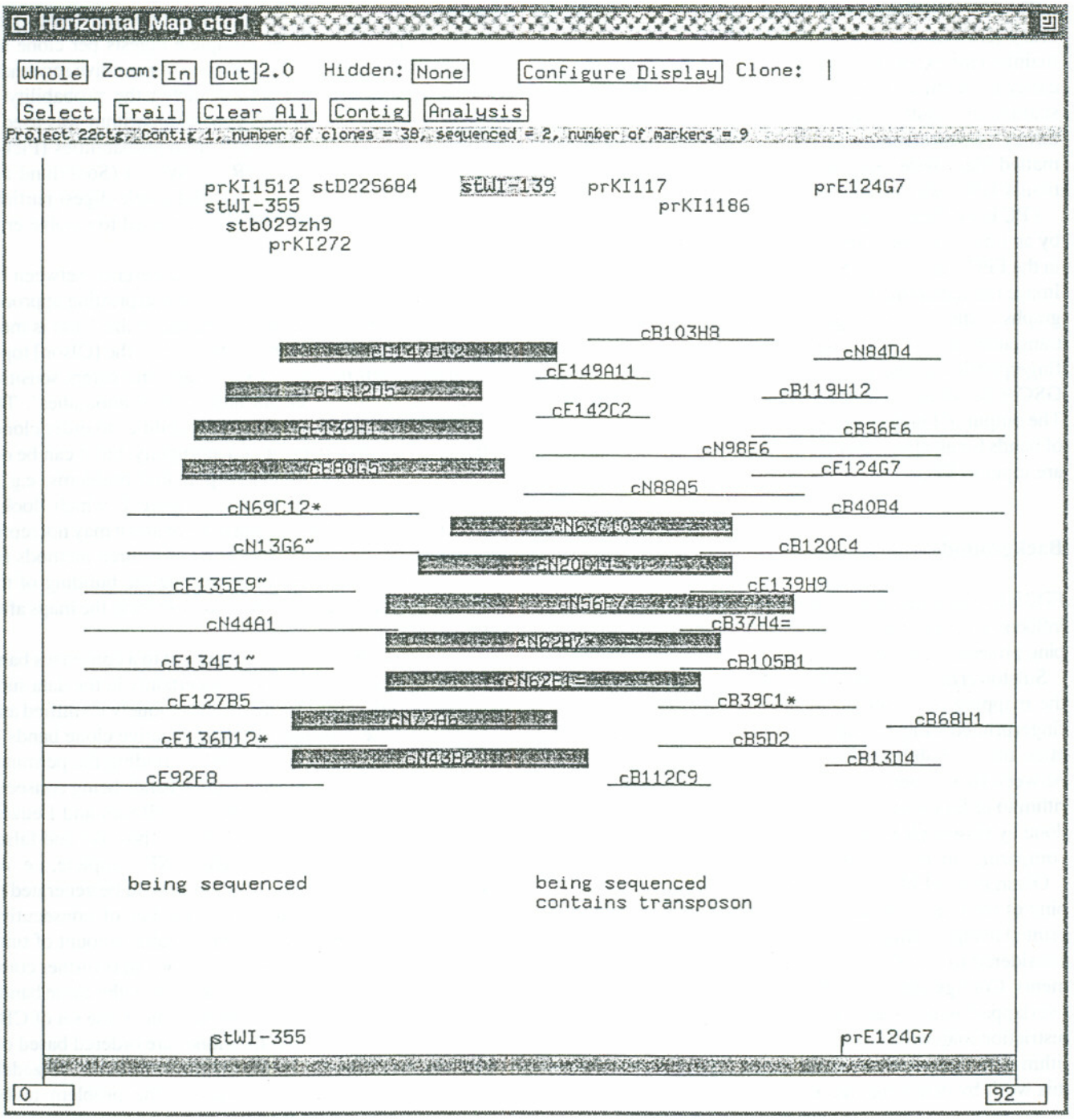


Fig. 1. The contig display shows clones as positioned by the user and/or CB routine. Each marker is positioned above the largest stack of clones to which it hybridizes. The rectangular box on the bottom shows the portion of the contig displayed. The 'anchors' along the bottom are markers selected by the user; all anchors are always displayed even when only a region of the contig is shown. When a marker or clone is picked, its 'friends' are highlighted. In this example, marker stWI-139 has been picked which highlighted all the clones which hybridized to it. The buried clones are shown, but can be hidden by toggling the 'None' button. The characters {*, =, ~} beside clone names indicate canonical, buried equal, and buried approximate, respectively.

The FPC windows are built with the ACEDB graphics library (Durbin and Thierry-Mieg, 1994); we have tried to maintain the exact look and feel of ACEDB so that users can easily use both systems simultaneously. Although FPC is a stand-alone system, it supports data flow with ACEDB through a set of functions that read and write ACEDB formatted files; these functions can easily be adapted for a particular laboratory's database.

FPC input files containing the bands must be generated first by an image-analysis program. The file format is as specified in the FPC manual. The Sanger Centre uses a program called Image (Sulston *et al.*, 1989) to scan and process the autoradiographs of the fingerprint gels; this program has recently been translated into C and upgraded (http://www.sanger.ac.uk/fingerprinting/imageprocessing). Image is also used by the GSC in St Louis for agarose gel data (complete fingerprints). The output of Image is one file per gel which contains the set of bands for all clones on the gel; these files, called band files, are input to FPC.

Background

FPC is a second-generation software project. Its design has been influenced by the software developed for three physical mapping projects: *C.elegans*, yeast and human chromosome 16.

Sulston *et al.* (1988) developed the CONTIG9 program for the mapping of *C.elegans* where cosmid clones have been fingerprinted using the approach of Coulson *et al.* It calculates the probability that the number of bands matched between two clones is a coincidence. It used a greedy algorithm to order the clones, although the majority of work was done by viewing lists of coincidence scores and interactively comparing fingerprints and positioning clones.

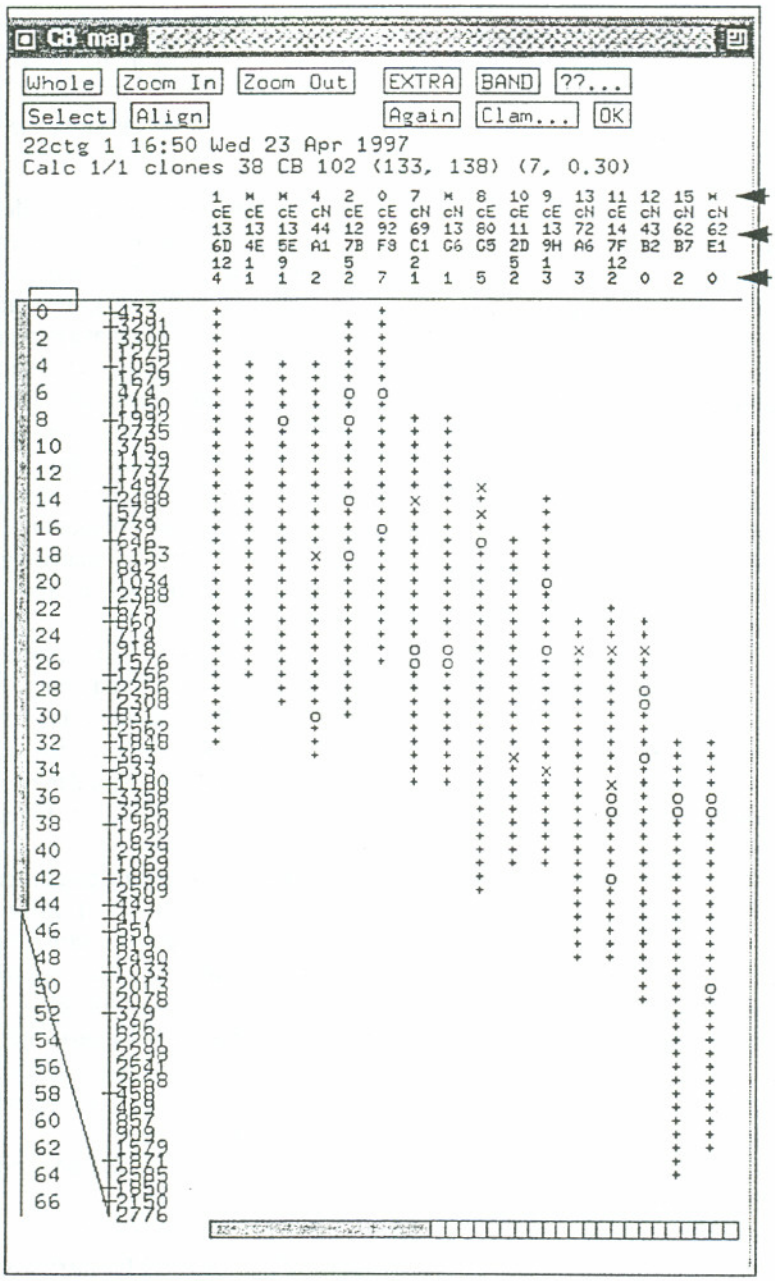
Olson *et al.* (1986) developed their software for the mapping of the yeast genome where lambda clones were fingerprinted using a complete restriction digest. Two clones were considered to overlap if they had five or more shared fragments. Contigs were created by clustering all the clones that overlapped with at least one other clone in the contig. A restriction map was built for a contig by using a greedy algorithm with backtracking. Gillett *et al.* (1995) have extended this work by detecting and repairing collapsed fragments.

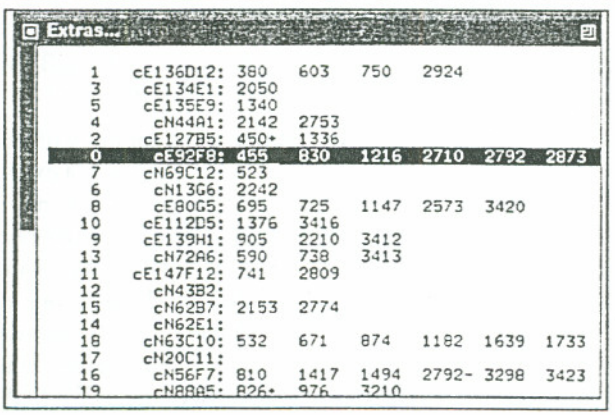
Stallings *et al.* (1992) fingerprinted cosmid-clones from chromosome 16 with three complete digests per clone and scored each fragment for the presence of repetitive sequences. Balding and Tomey (1991) calculated the probability of overlap using Bayes' rule. A genetic algorithm was used to order the clones based on the overlap probabilities (Fickett and Cinkosky, 1992). The GRAM system (Soderlund and Burks, 1994) was developed to build single-digest partially ordered restriction maps and was also used to resolve error in the contig maps.

Olson *et al.* (1986) summarize the difference between the first two approaches: 'The [Sulston] fingerprinting approach allows less extensive overlaps to be recognized and is more forgiving of minor errors in the data, while the [Olson] topological approach produces exact maps and offers sensitive detection of mapping errors and cloning anomalies'. The third approach of using overlap probabilities to order clones is the least sensitive to error and ambiguity, but it can be deceiving as it does not elucidate potential problems, e.g. it could produce a map as shown in Figure 1, which 'looks' correct regardless of the amount of error and it may not represent the true overlap structure at all. All three methods require human intervention, either during the building of the maps in the first two cases or to resolve error in the maps after the fact in the latter.

If the clones are ordered by alignment to a consensus band map, and if there is no error and uncertainty in the data such that the set of consensus bands can be uniquely identified and there are no false-positive and false-negative clone bands, a perfect assembly can be computed by finding the permutation of bands which results in all clone bands being consecutive. This can be solved in linear time (Booth and Leuker, 1978). Given a correct band set but false-positive and false-negative clone bands, the problem is NP-complete, i.e. all possible permutations of the band set have to be generated to find the order that maximizes the number of consecutive clone bands, and this takes an unacceptable amount of time for any realistically sized input. The problem is further complicated since a CB set must be inferred from the clone bands and, due to the uncertainties in measurement, the set of CBs will be an approximation. If the clones are ordered based on overlap probabilities, the problem is still difficult; Alizadeh *et al.* (1993) show two variations of the problem to be

Fig. 2. The CB map shows the consensus bands along the left; the marks delimit partially ordered groups. The top row of numbers is the position of the clone in an existing contig, an '*' indicates that the clone is buried in the first non '*' clone to the left. Beneath the position row is the clone name, followed by the number of extra bands; these are the bands that could not be placed in the CB map. Selecting the Extra.... button displays a window of the extra bands as shown on the bottom. Extra bands could be false positive, error in the map, or bands that will extend the map (no CB is made unless a consensus from at least two clones is present). Below the clone name is a column of characters {+, x, o} within the extents of the alignment of the clone to the CB map; these indicate whether the clone has a band matching a consensus band on the left. A '+' indicates a match, 'x' indicates the clone has a band that is greater than the tolerance but less than two times the tolerance, and 'o' indicates no match. Selecting the ?? button aligns the extra bands to any match to the left or right of a clone's alignment.


CB map
Whole
Zoom In
Zoom Out
EXTRA
BAND
??...
Select
Align
Again
Clam...
OK
22ctg 1 16:50 Wed 23 Apr 1997
Calc 1/1 clones 38 CB 102 (133, 138) (7, 0.30)
Position
Clone name
Number of extra bands


Extras...
cE136D12: 380 603 750 2924
cE134E1: 2050
cE135E9: 1340
cN44A1: 2142 2753
cE127B5: 450+ 1336
cE92F8: 455 830 1216 2710 2792 2873
cN69C12: 523
cN13G6: 2242
cE80G5: 695 725 1147 2573 3420
cE112D5: 1376 3416
cE139H1: 905 2210 3412
cN72A6: 590 738 3413
cE147F12: 741 2809
cN43B2:
cN62B7: 2153 2774
cN62E1:
cN63C10: 532 671 874 1182 1639 1733
cN20C11:
cN56F7: 810 1417 1494 2792- 3298 3423

NP-hard and Golumbic *et al.* (1994) show two simplified versions to be NP-complete. Therefore, approximation algorithms must be used; the less error there is in the data, the better the approximation algorithm performs.

The CB algorithm

FPC's fast assembly algorithm generates an approximate CB map as follows. (i) Compare the bands of each pair of clones and calculate the probability that the number of shared bands is a coincidence. (ii) Bury clones that are exact or a close match to another clone. (iii) Build a consensus band map using a hybrid greedy/stochastic algorithm to build the initial map, and then greedily extend the map. A CB map is shown in Figure 2, the '–' next to some of the CBs delimit groups of bands. Groups are ordered, but bands within a group are not, i.e. there is not enough information to determine the order of bands within a group. The o's indicate false negatives but can be caused from errors in the map building.

The following details the three steps.

Compare two clones

The bands for a pair of clones are compared and the number of matched bands (i.e. within the tolerance) M is calculated by sorting the two lists in descending order and stepping down the lists comparing bands. The relative significance of M depends on the numbers of bands that each clone has, so FPC uses a coincidence score equal to the probability of observing M matches given the observed band counts. Thus, a low probability score implies that the clones most likely overlap. Note that FPC only uses these scores to rank the clone overlaps and it does not require the clone overlap score to be interpretable as a probability. Only coincidence scores below a threshold are considered by FPC. However, if we do interpret the scores as probabilities, then for N clones, we make $C = N(N-1)/2$ comparisons and the most extreme probability score expected by chance will be about $1/C$, which is the guide value for the threshold. The user can choose between the following two equations for computing the coincidence score.

Equation 1. CONTIG9 (Sulston *et al.*, 1988) calculates the probability of M matches being a coincidence as follows. Let nL, nH be the minimum and maximum numbers of bands for the two clones ($nL < nH$). Clearly, M can never be greater than nL. Assuming all bands are equally likely, and for a fixed tolerance t (default 7), the approximate probability b of a matching band is $2t/gellen$ where $gellen$ is the gel length (default 3300). Therefore, the probability that none of the nH bands match with a given band is $p = (1-b)^{nH}$, and the number of matches M will be distributed as Binomial $B(nL,p)$. Consequently, the probability of observing at least M matches is:

$$\sum_{m=M}^{nl}\left[\binom{nL}{m}((1-p)^m p^{nL-m})\right] \qquad (1)$$

For variable tolerance, b is calculated as described at the end of this section.

Equation 2. We have found that the following simple statistical model gives more conservative overlap probabilities for a given threshold. Suppose that there are B possible sizes that a band can have (e.g. $B = 3000$), and that each possible value is equally likely. We observe that a particular clone has exactly n bands ($n < B$), represented as a binary vector of length B containing n ones and $B–n$ zeros. We model these data as being a sample from the population of binary B vectors in which each element is 1 with probability n/B, independently of the others. Consequently, the expected number of positives in a sample from this population is n.

Now consider the number of matching bands for two non-overlapping clones A and B with n_A and n_B bands, respectively (there is no requirement that $n_A < n_B$). If the fingerprints for the two clones were sampled independently from two different populations, as described above, then the probability that a particular band occurs in both clones is $q = (n_A/B)(n_B/B)$. The distribution of the number of matching bands is then Binomial $B(p,B)$, and the probability of observing M or more matching bands is as for equation (1), with n_L replaced by B and p by q.

The probability of a matching band becomes more complicated when a constant tolerance t is allowed for two bands to be considered a match. It is necessary to take into account the way matches are counted, e.g. when two bands for clone A are within t of the same band for clone B. If such a match counts only once, then a good approximation to the probability q_t of a matching band at size S is the sum of:

prob(A has band at S, B has no band in $[S–t...S–1]$, but has band in $[S...S + t]$) +
prob(B has band at S, A has no band in $[S–t...S–1]$, but has band in $[S...S + t]$ –
prob(A and B have bands at S)

The third term is subtracted to avoid counting that event twice. Hence:

$$q_t = (n_A/B)(1-(1-n_B/B)^{t+1})(1-n_B/B)^t + (n_B/B)(1-(1-n_AB)^{t+1})(1-n_A/B)^t - q$$

and the distribution of the number of matching bands is Binomial $B(q_t,B)$. This formula is not exact because it does not cover cases in which a cluster of bands are within t of each other. Nevertheless, this formula has been tested by simulation and shown to be sufficiently accurate. Table 2 shows example probabilities for the two formulas for different nL, nH and M. For moderate scores, the two equations give similar answers, but in more extreme cases equation (1) is consistently smaller than equation (2). We find that either formula works well for ranking clone overlaps.

Table 2. Coincidence scores. The variables *nL* and *nH* are the number of bands in the two clones being compared, and *M* is the number of matched bands. The tolerance is 7 and *gellen* is 3300. *Eq1* and *Eq2* are the scores produced by the corresponding equations (see the text)

nL	nH	M	Eq1	Eq2	nL	nH	M	Eq1	Eq2
10	10	5	3e-05	9e-05	10	30	10	6e-10	9e-07
20	20	10	1e-06	1e-05	10	40	10	9e-09	9e-06
40	40	20	4e-07	8e-06	10	50	10	7e-08	5e-05
60	60	30	3e-06	3e-05	10	60	10	3e-07	2e-04
5	10	5	1e-07	4e-06	30	30	20	3e-12	2e-09
10	20	10	1e-11	3e-08	30	40	20	5e-10	2e-07
20	40	20	8e-17	3e-10	30	50	20	2e-08	3e-06
30	60	30	4e-20	8e-11	30	60	20	3e-07	3e-05

Bury clones

If all the bands of one clone are a subset of another's, the algorithm buries the shorter clone in the longer clone (this is a concept adopted from CONTIG9, where the parent clone is called 'canonical'). A clone can be buried with a status of 'exact' if the bands are exactly the same or 'approximate' if the number of matched bands is within a user-defined limit. Buried clones can be suppressed from the contig display in order to reduce the amount of output. Also, they identify good clones, as when one set of bands is the exact subset of another, those bands are probably good and not just coincidental. This fact is used in the CB algorithm, as stated below.

Build the CB map

The clones are selected for incorporation into a CB map by using a priority search of a weighted adjacency matrix (Sedgewick, 1988) where two clones are adjacent if they have a coincidence score that is below the user-defined cut-off (i.e. a high probability of overlap). The matrix is implemented as a sparse matrix for time and space efficiency; i.e. for each clone C_i, a list Adj(C_i) is created that contains all the clones adjacent to it. For each $C_j \in$ Adj(C_i), the weight w is set to ∞ if clone C_j is buried in clone C_i, otherwise, w is set to the exponent of the coincidence score for C_i and C_j.

The matrix is traversed using a priority first search (PFS) which generates one or more connected components (a connected component corresponds to a contig). The PFS uses a priority queue. When C_i gets placed on the queue, V_i is set to 1 (this indicates that it has been 'visited'). The function PQinsert(C_j, w) checks to see if C_j is on the queue and if so, its priority is upgraded if w is greater than its current priority; if C_j is not on the queue and V_j is 0, it is added to the queue with priority w. The function PQremove() removes from the queue the clone with the highest priority and returns it.

When C_i is aligned to the map, L_i and R_i are set to the left and right coordinate, respectively; these can be updated on a left or right extension of the CB array. The arrays L and R are initialized to $+\infty$ and $-\infty$ respectively. The array CB stores the CBs. The array G stores pointers into the CB array to delimit the groups.

1. Set C1 = the clone that has the most high overlaps.
   For each $C_j \in$ Adj(C1) do PQinsert(C_j, w).
2. Set C2 = PQremove().
   For each $C_j \in$ Adj(C2) do PQinsert(C_j, w).
   Initialize the CB map with the bands that are shared between C1 and C2.
3. Until all clones are gone from the queue:
   Set C = PQremove().
   For each $C_j \in$ Adj(C) do PQinsert(C_j, w).
   - Align its bands with the current CB map within the window $[l, r]$
     where $l = \min\{L_j : C_j \in \text{Adj(C)}\}$ and $r = \max\{R_j : C_j \in \text{Adj(C)}\}$.
   - If the alignment positions C near the left or right end of the map,
     try to extend it to the left or right. An extension creates a new group.
   - If the CB map has X or more bands and S=0, shuffle and realign and set S=1.
4. If there are any clones C_i with V_i =0, go to 1 and start a new CB map.

Fig. 3. An outline of the CB algorithm.

An outline of the algorithm is shown in Figure 3 (– items are described more fully below).

The CB algorithm carries out a two-pass alignment; the first pass uses $\pm t$ (the tolerance) for determining whether a band matches with a CB band. The second pass fills in gaps in the alignment using $\pm 2t$. Adding this rule greatly improved the results as (i) the bands are often outside the tolerance, but to use $2t$ on a single pass creates too many bad bands, and (ii) the CB map uses an average of all the clone bands included as positive, and this average can drift. The bands that are within twice the tolerance are marked with an 'x' in the CB map.

Aligning the bands to the current CB map is not a straightforward alignment because of the groups. An alignment must take into account that the bands can match anywhere in the left and right group, whereas it has to have an almost exact match to the internal groups. Given groups $G_1 < G_2 .... < G_n$, and an alignment to the interval $[b_i, b_j]$ where $b_i \in G_i$ and b_j

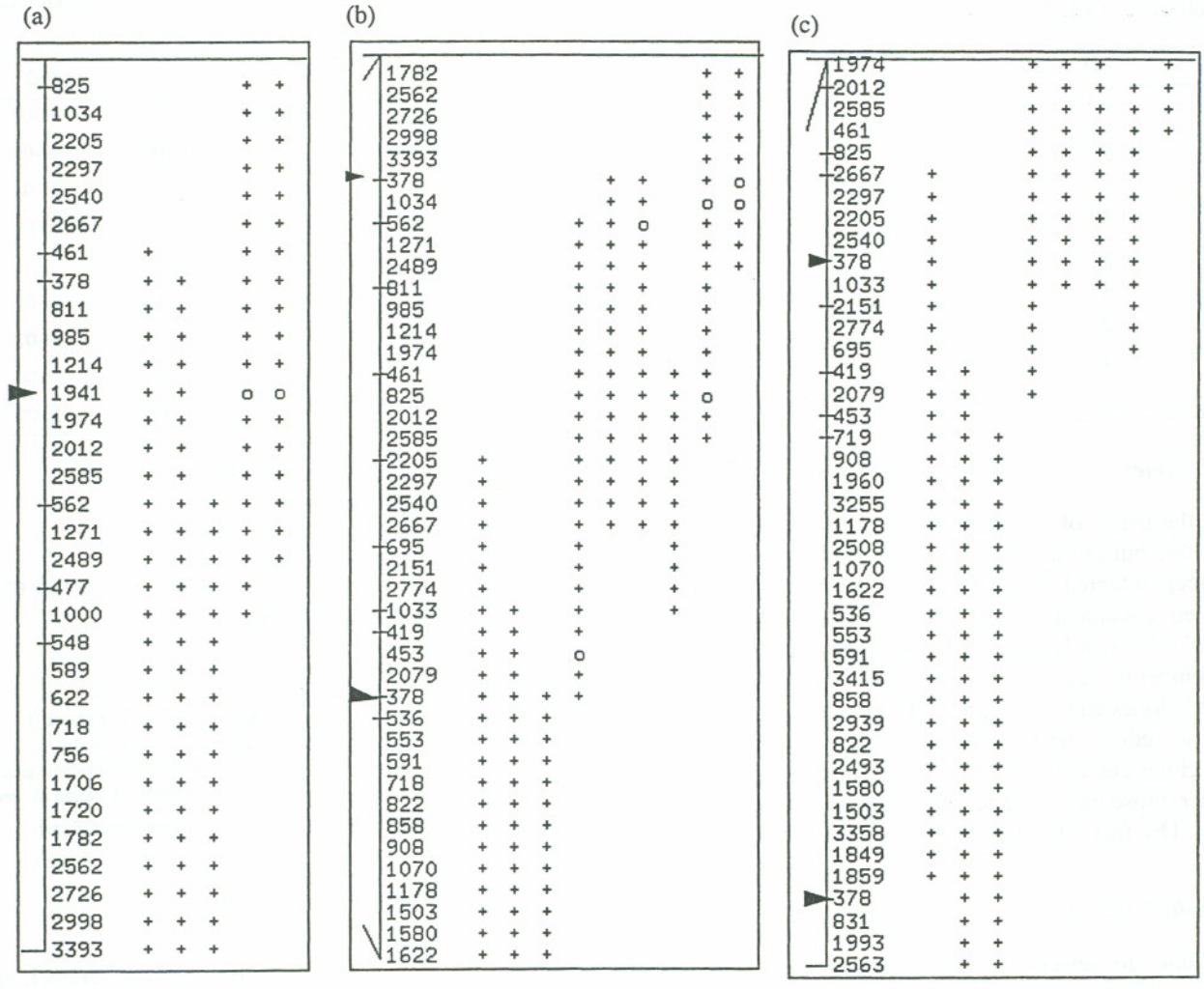


Fig. 4. (**a**) Misplaced fragment. Band 1941 was assigned to a group based on the constraints of the first three clones; given the added information of the next two clones, the band should be at the low end of the CB map. The shuffle algorithm corrects this error (correct solution not shown). (**b**) Collapsed fragment. The second occurrence of band 378 is a collapsed band. The fourth clone should have a '+' with the first occurrence of 378 instead of the second. (**c**) The shuffle and realign corrected the error. [Not all of the CBs are shown in (b) and (c) as the zoom feature was used to see the results better.]

$\in G_j$, $i \leq j$, the number of matched bands M and non-matched bands NM are counted for each group. The score for the alignment is computed by: $N(G_i) + N(G_j) + \Sigma_{i<k<j} (N(G_k) - NM(G_k))$. The best alignment is incorporated into the map. If $i \neq j$ and $i \neq 1$, the matched bands in G_i are rolled to the right and a new group is made of the non-matched bands. If $i \neq j$ and $j \neq N$, the matched bands in G_j are rolled to the left and a new group is made of the non-matched bands. If $i = 1$ or $j = N$, a heuristic determines whether the alignment should result in rolled bands or an extension to the CB map. (For example, what if an alignment matches a significant number of bands to the right of G_1, matches three out of six bands in G_1, and can be extended by three bands, should it roll or extend?)

After the map has a few groups, the positions of new ones are well defined due to the topological constraints, e.g. if a clone aligns to the left end of the CB map, any new bands must obviously go on the left end, but on the creation of initial groups, they can end up in the wrong order due to the lack of constraints. Or, as shown in Figure 4, a band may be in the wrong group. The following technique generally fixes misplaced bands or groups. The algorithm carries out a stochastic shuffle of the bands after a user-defined number of bands (default 50) have been added, then the clones are realigned. The stochastic shuffle is the same algorithm as described in Soderlund and Dunham (1995); it re-orders the CB bands to maximize the number of consecutive clone bands. The clones that have already been incorporated into the map

are realigned to the newly ordered CB bands, and CBs are added and deleted as necessary. If any bands are removed, the shuffle and realign is executed again.

Gillett *et al.* (1995) describe the problem of fragment collapsing ' ... when two different genomic fragments of approximately the same length and occurring in the digestion of two different overlapping clones are incorrectly identified as representative of a single genontic fragment.' This is illustrated in Figure 4 where band 378 occurs twice, and the occurrence of 378 in the fourth clone is assigned to the wrong one. This causes a ripple effect, i.e. some of the error (i.e. o's) is caused by this one collapse. This problem is often solved by the shuffle and realigned when it occurs in the initial building of the map, as shown in Figure 4.

If misplaced or collapsed bands are not resolved by the shuffle or if they are not prevented from the topological constraints, then they often manifest themselves as a consensus band with a few positives and many negatives. That is, a consensus band is created and yet newly aligned clones which cover the band consistently do not match. This phenomenon also occurs due to partial fingerprint data. The algorithm checks for this situation after each new alignment, and if a band has more mismatches than matches it is removed.

If a good position for a clone cannot be found [i.e. with an alignment score BD where B is the number of clone bands and D is a user-defined variable (default 0.5)], the clone is aligned to the consensus map but no groups are created except if it can extend the map. The clone is listed with an 'A' above it in the CB map display. Originally, poorly aligned clones where rejected, but this resulted in two problems: (i) a rejected clone could extend the map, i.e. provided the bridge between two subcontigs; (ii) it is desirable to include all clones above the cut-off and let the user decide what to reject. The CB algorithm can then be re-run for better results after removing clones.

If problems still occur, they can often be avoided by starting with a different clone. The user can request a new solution any number of times, each time it starts with a different clone which produces an alternative solution. Additionally, the user can refine regions of the contig by running the algorithm on selected sets of clones.

In summary, the algorithm uses a hybrid of greedy, stochastic and heuristic approaches to try to reduce error. Different heuristics were tried (such as alternative scoring schemes, priority for adding clones, and values for the parameters) in order to find the heuristics that most consistently remove error without adding new error. Given a set of 36 clones with perfect fingerprints, the algorithm created a perfect map, but with real data, the clones added on the right end of the CB map usually have increasingly poor alignments due to the following: (i) any error in the map causes a ripple effect in that it creates more error as clones are added and (ii) the algorithm greedily adds the best clones first, i.e. with the best overlap scores. In practice, the user must view the CB map and determine what clones to remove. After removing bad clones, the algorithm should be run again as it will obviously produce a better solution with a reduction in poor fingerprints.

Variable tolerance. For complete digest data, a variable tolerance is used when comparing bands. The user has a choice of using either (a) a variable tolerance expressed as a fixed proportion of the band sizes, i.e. two bands $b1$ and $b2$ may be the same if $| b1 - b2 | < t * 0.001 * (b1 + b2)/2$, or (b) a file of uncertainty values with linear interpolation. If a single tolerance value is used, it is input as an integer; for equation (1), b is calculated as described for fixed tolerance except that the *gellen* is 8000. The $2t$ in the numerator does not reflect the variable tolerance, but it is better for ranking to use a constant value. These values were tested by running the CB algorithm on a data set from GSC in St Louis where they had a set of bands (migration values) for 11 608 clones and a set of fragment lengths for the same clones. Equation (1) produces similar values for the two data sets.

Implementation

The user's manual (Soderlund and Longden, 1996) provides a complete description of all FPC functions. Soderlund *et al.* (1997) give a step-by-step description of how the Sanger Centre uses FPC to select sequenc-ready clones. This section will briefly review the salient interactive features of FPC.

Database

The FPC database consists of two types of flat files: an ACEDB style file which contains all the information except for the bands, and a file containing all the bands. When the user executes the Update.cor function, both files are updated with the new data from the band files; the band files are then moved to a special directory to be removed by the user. FPC does not support concurrent write, but it does provide a Write Lock so that a user can lock the database while making changes, and FPC does not allow a write if the file has been written by another user since the last read in a session.

Tolerance

Figure 5 shows the main FPC window. The Configure window allows the user to set the tolerance to variable (for agarose gels) and select equation (2) [default is equation (1)]. On the Analysis window, the user can set the Tolerance, the Cutoff is the threshold for determining when two clones overlap, and the Diff Bury value for determining how many bands must match for a bury. These values are saved in the FPC file.

Edit

FPC has three classes: contigs, clones and markers. Contigs are displayed as maps (see Figure 1) and contain clones and

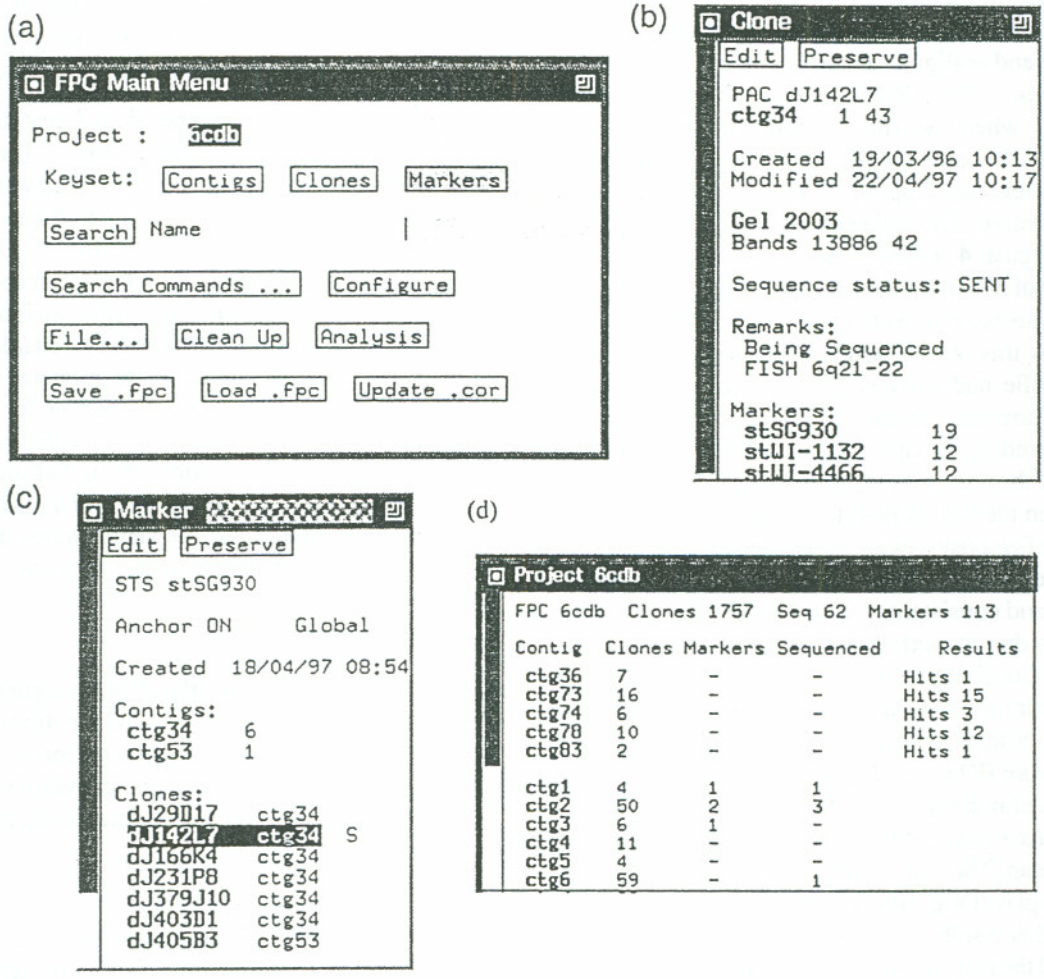


Fig. 5. (**a**) The FPC main window. (**b**) A clone text window. (**c**) A marker text window. (**d**) The Project window with results from comparing all the singletons against the contigs; the 'Hits' are the number of clones with a significant overlap with one or more clones in the contig.

markers; a contig can be edited by moving, adding and deleting clones. Clones are displayed as text windows (see Figure 5); they can have remarks, markers, sequence states and buried clones. Markers are displayed as text windows; each marker must have a positive hybridization to at least one clone and can belong in multiple contigs if attached to multiple clones. Editing capabilities are available for clones and markers.

Search

The clones can be searched by name, date, gel, contig or remark. The markers can be searched by name or date. The results are displayed in a keyset.

Gels

The new Image program has a function to write a gel image file for each gel (F.Wobus and R.Durbin, personal communications). If these files exist, the user can request the gel image from multiple clones (from different gels) to be displayed in a window (see Figure 6). A clone can have more than one fingerprint and the gel images can be displayed together in the gel image window.

Create contigs and analyle existing contigs

The CB map algorithm can be run in the following three situations. (i) It is run on all the singletons in the database (i.e. clones not assigned to contigs). The user can 'OK' all the CB maps which creates a contig for each map, instantiates the burying of clones, and assigns coordinates to the clone corresponding to its alignment with the map. (ii) It can be run on an existing contig so that the user can determine clones to cancel and coordinates to change. (iii) It can be run on a selected set of clones picked from a contig display.

There are functions to aid the user in adding new clones to existing contigs which write results to a log file. (i) A keyset

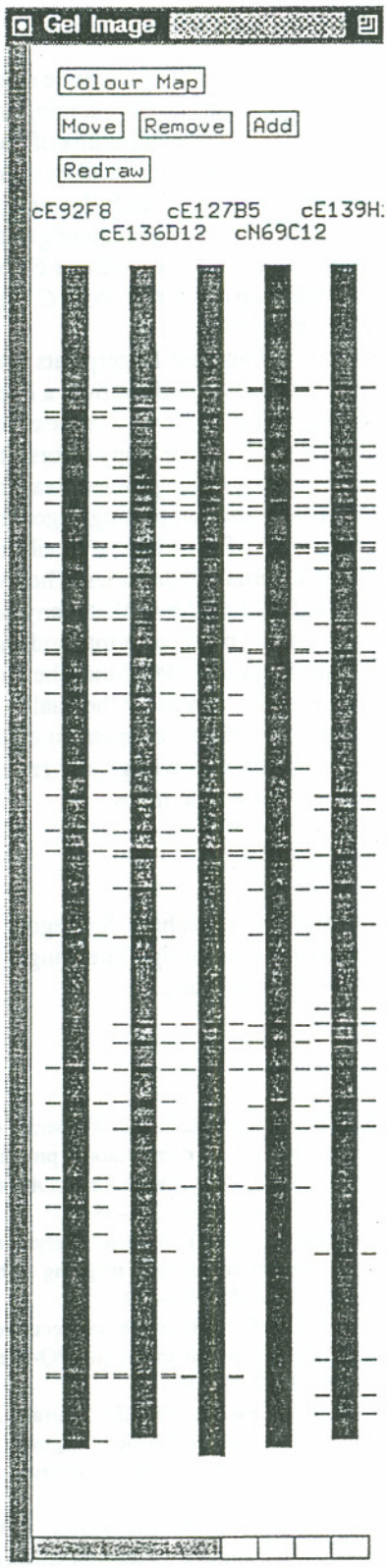


Fig. 6. The Gel Image window allows the user to view multiple gel lanes from different gels in one window. The ticks beside the images indicate the bands called positive.

of clones can be compared against all clones in the database. (ii) The singletons can be compared against all contigs; besides writing to a file, a window is displayed with all the contigs that have one or more clones overlapping with the one or more singletons, as illustrated in Figure 5. (iii) For a displayed contig, the keyset can be compared against all clones in a contig, and the user can step though the clones that match the contig and the matched contig clones are highlighted for each; the user can then request that a clone be added to the contig. There are also functions for comparing ends of contigs to determine contigs to merge; the user can then request that two contigs be displayed together so that the fingerprints of the end clones can be compared to determine if and by how much to merge the contigs, and then accept or reject the merge.

Interactions between windows occur, e.g. clicking on a clone in the CB map or gel image window highlights it in the contig window, clones from a keyset can be selected on a contig window or a keyset can be created from the selected clones, clicking a clone will highlight the markers contained within it, etc. Finally, there are many ways to analyse existing contigs and individual clones within a contig; for example, when a user is considering picking a clone to be sent for sequencing, the clone is compared with the rest of the database to ensure the uniqueness of the clone.

Discussion

FPC is a system built to cluster a large number of fingerprinted clones into contigs and assign coordinates to the clones. The probability of coincidence is calculated to determine the clones that have a high probability of overlap. Clones are clustered into contigs based on the overlap score. A CB map is calculated to aid positioning of clones. A CB map is a low-resolution restriction map, i.e. it does not resolve all error. FPC has a function to write a contig into a GRAM formatted file so that a detailed partially ordering restriction map of a contig can be built with GRAM (Soderlund and Burks, 1994).] FPC has a complete set of editing functions so that the user can refine the maps by removing, adding or moving clones.

In order to get a minimum number of false-positive overlaps, the coincidence score cut-off must be stringent, but then many true overlaps are ignored. The detailed displays of the fingerprints (e.g. the CB map and gel image display as shown in Figures 2 and 6, respectively) aid the user in finding the small valid overlaps. These displays also aid the biologist in determining clones that have the appropriate amount of overlap and would be good candidates for sequencing; these clones can be tagged by setting the sequence status flag for a clone and all clones with the sequence state set will be highlighted for a contig upon request.

FPC allows the incorporation of markers by either reading them from an external file or by adding them through the FPC editor. We were originally planning on using the marker data in conjunction with the fingerprints to determine clone order, but we have found that keeping the two analyses separate has been valuable for finding error in either type of data. By having the markers displayed with the clones, errors are easily found. FPC allows a set of markers to be defined as the 'global marker' order. Contigs can be listed by their global marker order which aids in determining contigs to merge; in other words, the global markers provide a scaffold. Markers, global markers and sequence-ready clones are discussed further in Soderlund *et al.* (1997).

Currently, the largest Sanger Centre mapping project using FPC mapped 17 738 clones for chromosome 22, which represents a 12× coverage of a 56 megabase chromosome. The number of contigs is 323 with an average of 43 clones per contig. The largest contig is 1716 clones with a total length of 1169 bands. The number of markers is 863. Several biologists worked on assembling and merging the contigs by each taking a region of the chromosome (S.Gregory, personal communications).

A short-coming we have found with FPC V2.5 is the awkwardness of processing clones that have fingerprints that give a high probability of being in a particular contig, but the bands are not good enough for accurate positioning. What we want in a contig is a depth of four or five of very good clones (e.g. clones that are canonical to exact or approximate matches indicate they have good bands). We want to keep other 'poorer' clones with the contig, but as a third-class set which are usually ignored. We have recently added a third type of burying called 'pseudo' to address this issue. A second problem is that the routine for locating individual clones in a contig only positions them on the left end of the clone in which it has the best overlap and the user must refine the position. A third problem is that there is not enough automation for merging contigs.

Future

An algorithm should be written that uses the CB algorithm to build small solid sub-contigs and then merges them based on residual band overlap and marker data. Contigs with a maximum depth of four or five should be created and all other clones for a region should be buried (using the new 'pseudo' status when necessary). This would build better maps and make it plausible to automate the addition of new clones to existing contigs.

An algorithm should be written to compare the results of two CB maps that have the same set of clones but are started with a different clone. If the order changes greatly, the quality of the map is poor. The algorithm should try to identify the offending clones.

Conclusion

FPC has been in use by the Sanger Centre since the beginning of 1996. It has been used for the mapping of chromosomes 22, X and 6. It has been used with a mix of cosmids, fosmids, PACs and BACs. It is also being used by many other laboratories. For example, the GSC in St Louis has been using it with complete digest data for the mapping of fosmid, cosmid, BAC, PAC and YAC clones on human chromosome 7 and the whole genome map of the nematode *C.briggsae* (M.Marra, personal communications).

Using restriction fragment fingerprints for building contigs has been a technique used for over a decade. There has been speculation that it is not very successful as contigs are hard to build and tend to have many errors. It is the authors' contention that the root of the problem was the lack of good interactive software and supporting algorithms. Although CONTIG9 was successful, its lack of flexibility probably reduced its use by external laboratories. The overwhelmingly good response to FPC has shown that many laboratories still desire to build contigs by fingerprints and there has been a real need for a program like FPC. Since the quality of the CB maps increases with the increase in the quality of the data, the biologists' efforts should now be spent on getting the highest quality fingerprints and this will greatly reduce the interactive time needed to assemble maps.

Acknowledgements

We would like to thank I.Dunham, S.Gregory and R.Durbin for comments on this manuscript. The Sanger Centre is supported by the Wellcome Trust.

References

Alizadah,F., Karp,R., Newberg,L.A. and Weisser,D. (1993) Physical mapping of chromosomes: A combinatorial problem in molecular biology. In *Proceedings of the Fourth Annual ACM_SIAM symposium on Discrete Algorithms*, pp. 371–381.

Balding,D. and Torney,D. (1991) Statistical analysis of DNA fingerprint data for ordered clone physical mapping of human chromosomes. *Bull. Math. Biol.*, **53**, 853–879.

Booth,S. and Lueker,G. (1976) Testing the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Comput. Syst. Sci.*, **13**, 335–379.

Bransomb,E., Slezak,T., Pae,R., Galas,D., Carrano,A. and Waterman,M. (1990) Optimizing restriction fragment fingerprinting methods for ordering large genomic libraries. *Genomics*, **8**, 351–366.

Coulson,A., Sulston,J., Brenner,S. and Karn,J. (1986) Toward a physical map of the genome of the nematode *C.elegans*. *Proc. Natl Acad. Sci. USA*, **83**, 7821–7825.

Durbin,R. and Thierry-Mieg,J. (1994) The ACEDB Genome Database. In Subai,S. (ed.), *Computational Methods in Genome Research*. Plenum Press, New York, pp. 45–56.

Fickett,J. and Cinkosky,M. (1992) A genetic algorithm for assembling chromosome physical maps. In Lim,H., Fickett,J., Cantor,C. And Robbins,R. (Eds), *The Second International Conference on Bioinformatics, Supercomputing, and Complex Genomic Analysis*. World Scientific, New Jersey, pp. 273–285.

Gillett,W., Daues,J., Hanks,L. and Capra,R. (1995) Fragment collapsing and splitting while assembling high-resolution restriction maps. *J. Comp. Biol.*, **2**, 185–205.

Gillett,W., Hanks,L., Wong,G., Yu,J., Lim,R. and Olson,M. (1996) Assembly of high-resolution restriction maps based on multiple complete digests of a redundant set of overlapping clones. *Genomics*, **33**, 389–408.

Golumbic,M., Kaplan,J. and Shamir,R. (1994) On the complexity of physical mapping. *Adv. Appl. Math.*, **15**, 251–261.

Gregory,S., Soderlund,C. and Coulson,A. (1996) Contig assembly by fingerprinting. In Dear,P. (ed.), *Genome Mapping: A Practical Approach*. Oxford University Press, in preparation.

Kohara,Y., Akiyama,K. and Isono,K. (1987) The physical map of the whole *E.coli* chromosome: application of a new strategy for rapid analysis and sorting of a large genomic library. *Cell*, **50**, 495–508.

Olson,M., Dutchik,J., Graham,M., Brodeur,G., Helms,C., Frank,M., MacCollin,M., Scheinman,R. and Frank,T. (1986) Random-clone strategy for genomic restriction mapping in yeast. *Proc. Natl Acad. Sci. USA*, **83**, 7826–7830.

Sedgewick,R. (1988) *Algorithms*. Addison-Wesley, Redding, MA.

Soderlund,C., Torney,D. and Burks,C. (1993) Calculating shared fragments for the single-digest problem. In *Proceedings of the Twenty-sixth Hawaii International Conference on System Science*. IEEE Computer Society Press. Vol. 1, pp. 620–633.

Soderlund,C. and Burks,C. (1994) GRAM and *genfragII*: Solving and testing the single-digest partially-ordered restriction map problem. *Comput. Applic. Biosci.*, **10**, 349–358.

Soderlund,C. and Dunham,I. (1995) SAM: A system for iteratively building marker maps. *Comput. Applic. Biosci.*, **11**, 645–655.

Soderlund,C. and Longden,I. (1996) *FPC V2.5: User's Manual*. Sanger Centre, Technical Report, SC-01–96.

Soderlund,C., Gregory,S. and Dunham,I. (1997) Sequence ready clones. In Bishop,M. (ed.), *Guide to Human Genome Computing*. Academic Press, in preparation.

Stallings,R. *et al.* (1992) Evaluation of a cosmid contig physical map of human chromosome 16. *Genomics*, **13**, 1031–1039.

Sulston,J., Mallet,F., Staden,R., Durbin,R., Horsnell,T. and Coulson,A. (1988) Software for genome mapping by fingerprinting techniques. *Comput. Applic. Biosci.*, **4**, 125–132.

Sulston,J., Mallett,F., Durbin,R. and Horsnell,T. (1989) Image-analysis of restriction enzyme fingerprint autoradiograms. *Comput. Applic. Biosci.*, 101–105.

Waterman,M. and Griggs,J. (1986) Interval graphs and maps of DNA. *Bull. Math. Biol.*, **48**, 189–195.