Parallel Computation of a Maximum-Likelihood Estimator of a Physical Map

Suchendra M. Bhandarkar,* Salem A. Machaka,* Sanjay S. Shete† and Raghuram N. Kota*

**Department of Computer Science, †Department of Statistics, The University of Georgia, Athens, Georgia 30602-7404*

Manuscript received September 1, 2000
Accepted for publication December 19, 2000

ABSTRACT

Reconstructing a physical map of a chromosome from a genomic library presents a central computational problem in genetics. Physical map reconstruction in the presence of errors is a problem of high computational complexity that provides the motivation for parallel computing. Parallelization strategies for a maximum-likelihood estimation-based approach to physical map reconstruction are presented. The estimation procedure entails a gradient descent search for determining the optimal spacings between probes for a given probe ordering. The optimal probe ordering is determined using a stochastic optimization algorithm such as simulated annealing or microcanonical annealing. A two-level parallelization strategy is proposed wherein the gradient descent search is parallelized at the lower level and the stochastic optimization algorithm is simultaneously parallelized at the higher level. Implementation and experimental results on a distributed-memory multiprocessor cluster running the parallel virtual machine (PVM) environment are presented using simulated and real hybridization data.

GENERATION of entire chromosomal maps has been a central problem in genetics right from its early years (Sturtevant 1913). These maps are central to the understanding of the structure of genes, their function, their transmission, and their evolution. Recent advances in molecular genetics have led to a wealth of DNA markers along a chromosome and also eased the process by which these markers can be assayed. Consequently, the focus of current research has shifted from data collection to the computational problem of map assembly.

Chromosomal maps fall into two broad categories—*genetic maps* and *physical maps*. Genetic maps represent an ordering of genetic markers along a chromosome where the distance between two genetic markers is related to their recombination frequency. Genetic maps are typically of low resolution, *i.e.*, 1–10 Mb (Lander and Green 1987). While genetic maps enable a scientist to narrow the search for genes to a particular chromosomal region, it is a physical map that ultimately allows the recovery and molecular manipulation of genes of interest. A physical map is defined as an ordering of distinguishable (*i.e.*, sequenced) DNA fragments called *clones* or *contigs* by their position along the entire chromosome where the clones may or may not contain genetic markers. The physical mapping problem is therefore one of reconstructing the order of clones and determining their position along the chromosome. A physical map has a much higher resolution than a genetic map of the same chromosome, *i.e.*, 10–100 kb (Brody *et al.* 1991). Physical maps have provided fundamental insights into gene development, gene organization, chromosome structure, recombination, and the role of sex in evolution and have also provided a means for the recovery and direct molecular manipulation of genes of interest (Prade *et al.* 1997).

Several techniques exist for generation of physical maps from contig libraries. These techniques are specific to an experimental protocol and the type of data collected, for example, mapping by nonunique probes (Alizadeh *et al.* 1995), mapping by unique probes (Alizadeh *et al.* 1994; Greenberg and Istrail 1995; Jain and Myers 1997), mapping by unique endprobes (Christof *et al.* 1997), mapping using restriction fragments (Fasulo *et al.* 1997; Jiang and Karp 1997), mapping using radiation-hybrid data (Ben-Dor and Chor 1997; Slonim *et al.* 1997), and optical mapping (Muthukrishnan and Parida 1997; Karp and Shamir 1998; Lee *et al.* 1998). Likewise, several computation techniques based on deterministic optimization and stochastic optimization have been reported in the literature in the context of physical mapping. Examples of stochastic optimization algorithms include simulated annealing (Cuticchia *et al.* 1992, 1993; Mott *et al.* 1993; Alizadeh *et al.* 1994, 1995) and the random cost algorithm (Wang *et al.* 1994a) whereas those of deterministic optimization algorithms include linear programming (Jain and Myers 1997), integer programming (Christof *et al.* 1997), integer linear programming with polyhedral combinatorics (Christof and Kececioglu 1999), and semidefinite programming (Chor and Sudan 1995). Various statistical analyses of the aforementioned physical mapping techniques have also been reported in the literature (Lander and Waterman 1988; Arratia *et al.*

Corresponding author: Suchendra M. Bhandarkar, Department of Computer Science, The University of Georgia, 415 Boyd Graduate Studies Research Ctr., Athens, GA 30602-7404.
E-mail: suchi@cs.uga.edu

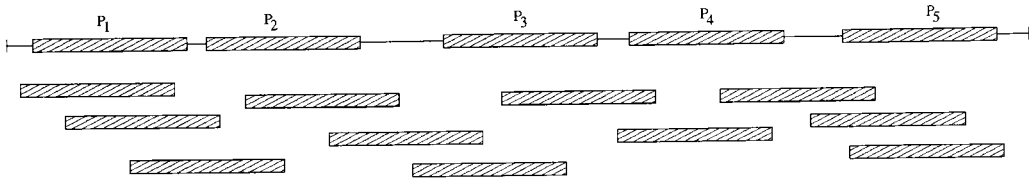

FIGURE 1.—An example of clone-probe ordering along a chromosome.

1991; ZHANG and MARR 1993; BALDING 1994; NELSON and SPEED 1994; XIONG *et al.* 1996; WILSON *et al.* 1997).

The physical mapping protocol: The physical mapping protocol essentially determines the nature of clonal data and the probe selection procedure. The physical mapping protocol adhered to in this project is the one based on *sampling without replacement* (FU *et al.* 1992). This protocol has been used successfully in physical mapping of *Aspergillus nidulans* (BRODY *et al.* 1991; PRADE *et al.* 1997), *Schizosaccharomyces pombe* (MIZUKAMI *et al.* 1993), and *Pneumocystis carinii* (ARNOLD and CUSHION 1997) and is currently being used in physical mapping projects of *A. flavus* and *Neurospora crassa* (AIGN *et al.* 2001; KELKAR *et al.* 2001) under the Fungal Genome Initiative (ARNOLD 1997; BENNETT 1997).

The protocol that generates the probe set $\mathcal{P}$ and the clone set $\mathcal{C}$ is an iterative procedure that can be described as follows. Let $\mathcal{C}^i$ and $\mathcal{P}^i$ be the clone set and the probe set, respectively, at the ith iteration. The initial clone set $\mathcal{C}^0$ consists of all the clones in the library whereas the initial probe set $\mathcal{P}^0 = \phi$. The clones in $\mathcal{C}^0$ are designed to be of the same length and to be overlapping so that each clone samples a fragment of the chromosome and the coverage of the entire chromosome is made possible. At the ith iteration a clone c is chosen at random from $\mathcal{C}^i$ and added to $\mathcal{P}^i$. Clone c is hybridized to all the clones in $\mathcal{C}^i$. The subset of clones $\mathcal{C}^c$ that hybridize to clone c are removed from $\mathcal{C}^i$ so that $\mathcal{C}^{i+1} = \mathcal{C}^i - \mathcal{C}^c$. Note that $c \in \mathcal{C}^c$ since a clone hybridizes to itself. The hybridization experiment entails extracting complementary DNA from both ends of a probe, washing the DNA over the arrayed plate, and recording all clones in the library to which the DNA sticks (*i.e.*, hybridizes). The above procedure is halted at the kth iteration when $\mathcal{C}^k = \phi$. The final probe set is given by $\mathcal{P} = \mathcal{P}^k$ and the clone set by $\mathcal{C} = \mathcal{C}^0 - \mathcal{P}^k$. In the absence of errors, the probe set $\mathcal{P}$ represents a maximal nonoverlapping subset of $\mathcal{C}^0$ since any clone that overlaps with a given probe would hybridize to one end of that probe and be effectively removed from consideration in future iterations of the aforementioned iterative procedure. Figure 1 depicts the probes and clones along the length of a chromosome where the clones and probes are numbered from left to right.

The clone-probe overlap pattern is represented in the form of a binary hybridization matrix H where the matrix element H_{ij} denotes the hybridization of the ith clone $\in \mathcal{C}$ to the jth probe $\in \mathcal{P}$. The matrix element $H_{ij} = 1$ if the ith clone $\in \mathcal{C}$ hybridizes to the jth probe $\in \mathcal{P}$ and $H_{ij} = 0$ otherwise. Table 1 shows the clone-probe hybridization data in the absence of errors resulting from the example depicted in Figure 1. If the probes in $\mathcal{P}$ were ordered with respect to their position along a chromosome, then by selecting from H a common overlapping clone for each pair of adjacent probes in $\mathcal{P}$ a minimal set of clones and probes that covers the entire chromosome (*i.e.*, a minimal tiling) could be obtained. Note that a common overlapping clone between two adjacent probes would hybridize to both probes. The minimal tiling in conjunction with the sequencing of each individual clone/probe in the tiling and a sequence assembly procedure that determines the overlaps between successive sequenced clones/probes in the tiling (KECECIOGLU and MYERS 1995) could then be used to reconstruct the DNA sequence of the entire chromosome.

In reality, the hybridization experiments are rarely error free. The hybridization matrix H could be expected to contain false positives and false negatives. The matrix element H_{ij} would be a false positive if $H_{ij} = 1$ (denoting hybridization of the ith clone with the jth probe) when in fact $H_{ij} = 0$. Conversely, H_{ij} would be a false negative if $H_{ij} = 0$ when in fact $H_{ij} = 1$. Other sources of error include chimerism wherein a single clone samples two or more distinct regions of a chromosome, deletions wherein certain regions of the chromosome are not sampled during the cloning process, and repeats wherein a clone samples a region of the chromosome with repetitive DNA structure. In this article, we

TABLE 1

An example of clone-probe hybridization data in the absence of errors

	Probes				
Clones	$\mathcal{P}_1$	$\mathcal{P}_2$	$\mathcal{P}_3$	$\mathcal{P}_4$	$\mathcal{P}_5$
$\mathcal{C}_1$	1	0	0	0	0
$\mathcal{C}_2$	1	1	0	0	0
$\mathcal{C}_3$	1	1	0	0	0
$\mathcal{C}_4$	0	1	0	0	0
$\mathcal{C}_5$	0	1	1	0	0
$\mathcal{C}_6$	0	0	1	0	0
$\mathcal{C}_7$	0	0	1	1	0
$\mathcal{C}_8$	0	0	0	1	0
$\mathcal{C}_9$	0	0	0	1	1
$\mathcal{C}_{10}$	0	0	0	0	1
$\mathcal{C}_{11}$	0	0	0	0	1

confine ourselves to errors in the form of false positives and false negatives. Since the clones (and probes) in the mapping projects that use the aforementioned protocol are generated using cosmids, which makes them sufficiently small (~40 kb), chimerism and deletions do not pose a serious problem. However, repeats do pose a problem but are not explicitly addressed here; rather they are treated as multiple isolated incidences of false positives.

In this article we present a maximum-likelihood (ML) estimator (Shete 1998; Kececioglu *et al.* 2000) that determines the ordering of probes in the probe set $\mathcal{P}$ and the interprobe spacings under a probabilistic model of hybridization errors consisting of false positives and false negatives. The estimation procedure involves a combination of discrete and continuous optimization, where determining the probe ordering entails discrete (*i.e.*, combinatorial) optimization, whereas determining the interprobe spacings for a particular probe ordering entails continuous optimization. We propose a two-level parallelization strategy for efficient implementation of the above estimator. The upper level consists of parallel discrete optimization using simulated annealing or microcanonical annealing whereas the lower level consists of the parallel conjugate gradient descent procedure. The resulting parallel algorithms are implemented on a distributed-memory multiprocessor cluster using the parallel virtual machine (PVM) environment (Sunderam 1990; Geist *et al.* 1994). Convergence, speed-up, and scalability characteristics of the parallel algorithms are examined and discussed.

MATERIALS AND METHODS

Cosmid libraries used to construct the physical map of *N. crassa* discussed in this article are described in Kelkar *et al.* (2001). The physical mapping data were generated by DNA hybridization described in Kelkar *et al.* (2001). Assignments of markers to physical and genetic maps was achieved by complementation, hybridization, and cosmid end sequencing as described in Kelkar *et al.* (2001).

MATHEMATICAL FORMULATION OF THE ML ESTIMATOR

In this section we present a brief synopsis of the ML estimator proposed in Kececioglu *et al.* (2000) and Shete (1998). The estimator reconstructs the ordering of probes in the probe set $\mathcal{P}$ and the interprobe spacings under a probabilistic model of hybridization errors consisting of false positives and false negatives.

The probe ordering problem can be formally stated as follows. Given a set $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ of n probes and a set $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ of k clones generated using the sampling-without-replacement protocol described earlier and the $k \times n$ clone-probe hybridization matrix H containing both false positives and false negatives with predefined probabilities, reconstruct the correct ordering $\Pi = (\pi_1, \pi_2, \ldots, \pi_n)$ of the probes and also the correct spacing $Y = (Y_1, Y_2, \ldots, Y_n)$ between the probes. The ordering Π is a permutation of $(1, \ldots, n)$ that gives the labels (indices) of the probes in left-to-right order across the chromosome. In the interprobe spacing vector Y, Y_1 denotes the space between the left end of the first probe P_{π_1} and the left end of the chromosome, and Y_i the spacing between the right end of probe $P_{\pi_{i-1}}$ and the left end of probe P_{π_i} (where $2 \le i \le n$). The spacing between the right end of probe P_{π_n} and the right end of the chromosome is given by $Y_{n+1} = N - nM - \Sigma_{i=1}^{n} Y_i$, where N is length of the chromosome and M is the length of each clone/probe. Note that our protocol requires that all probes and clones be of the same length.

The problem as stated above is ill posed in the precise sense as defined by Hadamard (1923). A problem is deemed well posed when its solution exists and is unique. A problem is ill posed when no solution exists or, if a solution does exist, it is not unique. In our case, the problem is ill posed since the underlying constraints do not imply a unique solution. Any probe ordering Π and any interprobe spacing vector Y that satisfies the requirements that $Y_i \ge 0$; $1 \le i \le n$ and $N - nM - \Sigma_{i=1}^{n} Y_i \ge 0$, constitute a valid solution. Hence the problem is formulated as one of determining a probe ordering and the interprobe spacings that maximize the likelihood of the observed hybridization matrix H given predefined probabilities for false positives and false negatives.

Mathematical notation: The mathematical notation used in the formulation of the maximum-likelihood estimator is given below:

N, length of the chromosome;
M, length of a clone/probe;
n, number of probes;
k, number of clones;
ρ, probability of false positive;
η, probability of false negative;
$H = ((h_{i,j}))_{1 \le i \le k,\, 1 \le j \le n}$; clone-probe hybridization matrix, where

$$h_{i,j} = \begin{cases} 1 & \text{if clone } \mathcal{C}_i \text{ hybridizes with probe } \mathcal{P}_j \\ 0 & \text{otherwise;} \end{cases}$$

H_i, ith row of the hybridization matrix;
$\Pi = (\pi_1, \ldots, \pi_n)$, permutation of $\{1, 2, \ldots, n\}$, which denotes the probe labels in the ordering when scanned from left to right along the chromosome;
$p_i = \Sigma_{j=1}^{n} h_{ij}$, number of 1's in H_i;
$P = \Sigma_{i=1}^{k} p_i$, total number of 1's in H, $Y = (Y_1, Y_2, \ldots, Y_n)$, vector of interclone spacings, where Y_i is the spacing between the right end of $P_{\pi_{i-1}}$ and the left end of P_{π_i} $(2 \le i \le n)$, and Y_1 is the spacing between the left end of P_{π_1} and the left end of the chromosome; and
$\mathcal{F} \subseteq \mathcal{R}^n$, set of feasible interprobe spacings $Y = \{Y_1, \ldots,$

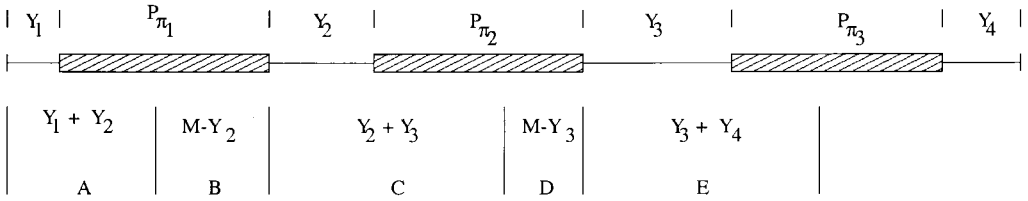

FIGURE 2.—Interprobe spacings: Case 1.

$Y_n\}$ such that $Y_i \geq 0$, $1 \leq i \leq n$, and $N - nM - \Sigma_{i=1}^{n} Y_i \geq 0$.

The model: Given a vector of interprobe spacings $Y = (Y_1, \ldots, Y_n)$, there are 2^{n+1} possible cases to consider depending on whether $0 \leq Y_i \leq M$ or $Y_i > M$, where $0 \leq i \leq n + 1$. Without loss of generality, we present the maximum-likelihood model for $n = 3$ and illustrate 3 of the $2^4 = 16$ possible cases.

Case 1: $0 \leq Y_1 \leq M$, $0 \leq Y_2 \leq M$, $0 \leq Y_3 \leq M$, and $0 \leq Y_4 \leq M$ as depicted in Figure 2.

Case 2: $0 \leq Y_1 \leq M$, $0 \leq Y_2 \leq M$, $0 \leq Y_3 \leq M$, and $Y_4 > M$ as depicted in Figure 3.

Case 3: $Y_1 > M$, $0 \leq Y_2 \leq M$, $0 \leq Y_3 \leq M$, and $0 \leq Y_4 \leq M$ as depicted in Figure 4.

In Case 1 above, if the left end of a clone falls in regions A, C, or E, then the clone will hybridize only with P_{π_1}, P_{π_2}, or P_{π_3}, respectively (Figure 2). Similarly, if the left end of a clone falls in region B or D, then the clone will hybridize with both P_{π_1} and P_{π_2} or P_{π_2} and P_{π_3}, respectively (Figure 2). In Case 2 above, if the left end of a clone falls in region E, then the clone will hybridize with only P_{π_3} and if it falls in region F, then the clone will not hybridize with any of the probes (Figure 3). Similarly, in Case 3 above, if the left end of a clone falls in region A, the clone will not hybridize with any of the probes (Figure 4). Therefore, from the above three cases it is easy to see that, in general, there will be three different types of regions, namely,

Type 1: *Both* region between probe P_{π_j} and $P_{\pi_{j+1}}$, for $j = 1, \ldots, n - 1$. An intervening clone hybridizes to both probes if its left end falls in this region.

Type 2: *Only* region of probe P_{π_j}, for $j = 1, \ldots, n$. A clone will hybridize to P_{π_j} only if its left end falls in this region.

Type 3: *None* region after probe P_{π_j}, for $j = 0, \ldots, n$. A clone will hybridize to no probe if its left end falls in this region. Here probe P_{π_0} is referred to as the beginning of the chromosome.

It can be shown that for $j = 1, \ldots, n - 1$,

Length of the *Both* region between probes P_{π_j} and $P_{\pi_{j+1}}$

$$= \begin{cases} 0 & \text{if } Y_{j+1} > M \\ M - Y_{j+1} & \text{if } Y_{j+1} \leq M \end{cases}$$

$$= M - \min(Y_{j+1}, M), \tag{1}$$

and for $j = 1, \ldots, n$,

Length of the *Only* region of probe P_{π_j}

$$= \begin{cases} Y_j + Y_{j+1} & \text{if } Y_j \leq M \text{ and } Y_{j+1} \leq M \\ M + Y_j & \text{if } Y_j > M \text{ and } Y_{j+1} \leq M \\ Y_j + M & \text{if } Y_j \leq M \text{ and } Y_{j+1} > M \\ 2M & \text{if } Y_j > M \text{ and } Y_{j+1} > M \end{cases}$$

$$= \min(Y_j, M) + \min(Y_{j+1}, M), \tag{2}$$

and for $j = 0, \ldots, n$,

Length of the *None* region after probe P_{π_j}

$$= \begin{cases} Y_{j+1} - M & \text{if } Y_{j+1} > M \\ 0 & \text{if } Y_{j+1} \leq M \end{cases}$$

$$= Y_{j+1} - \min(Y_{j+1}, M). \tag{3}$$

We assume that the left ends of the clones are uniformly distributed over the interval $[0, N - M]$; *i.e.*, the probes are uniformly distributed across the length of the chromosome. Therefore it can be shown that for $j = 1, \ldots, n - 1$, the probability P_{Both} that a randomly chosen clone will fall in the *Both* region of probes P_{π_j} and $P_{\pi_{j+1}}$ is given by

$$P_{\text{Both}} = \frac{M - \min(Y_{j+1}, M)}{N - M}; \tag{4}$$

for $j = 1, \ldots, n$ the probability P_{Only} that a randomly chosen clone will fall in the *Only* region of probe P_{π_j} is given by

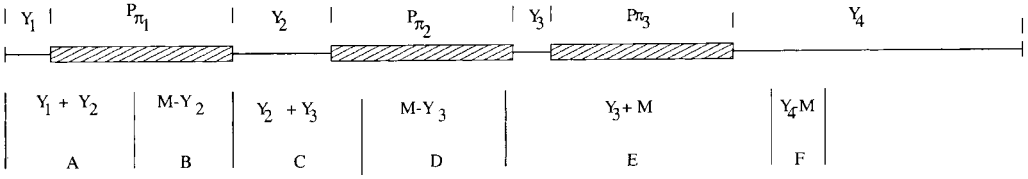

FIGURE 3.—Interprobe spacings: Case 2.

$$P_{\text{Only}} = \frac{\min(Y_j, M) + \min(Y_{j+1}, M)}{N - M}, \quad (5)$$

and for $j = 0, \ldots, n$ the probability P_{None} that a randomly chosen clone will fall in the *None* region after probe P_{π_j} is given by

$$P_{\text{None}} = \frac{Y_{j+1} - \min(Y_{j+1}, M)}{N - M}. \quad (6)$$

The conditional probability of observing a clonal signature H_i (*i.e.*, the ith row in the hybridization matrix H), given a probe ordering Π and an interprobe spacing vector Y, is given by

$$\begin{aligned} P(H_i \mid \Pi, Y) = &\sum_{j+1}^{n} P(H_i \mid \Pi, Y, O_{i,j}) P(O_{i,j} \mid \Pi, Y) \\ &+ \sum_{j=1}^{n-1} P(H_i \mid \Pi, Y, B_{i,j}) P(B_{i,j} \mid \Pi, Y) \\ &+ \sum_{j=0}^{n} P(H_i \mid \Pi, Y, N_{i,j}) P(N_{i,j} \mid \prod, Y), \quad (7) \end{aligned}$$

where $O_{i,j}$ is the event that the clone i will fall in the *Only* region of probe P_{π_j}, $B_{i,j}$ is the event that the clone i will fall in the *Both* region of probes P_{π_j} and probe $P_{\pi_{j+1}}$, and $N_{i,j}$ is the event that the clone i will fall in the *None* region after probe P_{π_j}.

For a given probe ordering Π and interprobe spacing vector Y, event $O_{i,j}$ implies that only $h_{i,\pi_j} = 1$, and all the remaining entries in row H_i should be equal to 0. In other words, $h_{i,\pi_j} \neq 1$ implies a false negative and a 1 in any other column position in the row H_i implies a false positive. That is,

$$h_{i,\pi_j} = \begin{cases} 0 & \text{with probability } \eta \\ 1 & \text{with probability } (1 - \eta), \end{cases} \quad (8)$$

and for $k = 1, \ldots, n$, where $k \neq j$,

$$h_{i,\pi k} = \begin{cases} 0 & \text{with probability } (1 - \rho) \\ 1 & \text{with probability } \rho. \end{cases} \quad (9)$$

Assuming that the false positive and false negative errors at different positions along the clonal signature H_i are independent of each other,

$$\begin{aligned} P(H_i \mid \Pi, Y, O_{i,j}) = &(1 - \eta)^{h_{i,\pi_j}} \cdot \eta^{(1 - h_{i,\pi_j})} \cdot \rho^{(p_i - h_{i,\pi_j})} \\ &\cdot (1 - \rho)^{(n-1) - (p_i - h_{i,\pi_j})}. \quad (10) \end{aligned}$$

Similarly, for a given probe ordering Π and interprobe spacing vector Y, event $B_{i,j}$ implies that only h_{i,π_j} and $h_{i,\pi_{j+1}}$ should be equal to 1 and all the remaining entries of row H_i should be equal to 0. That is,

$$h_{i,\pi_j} = \begin{cases} 0 & \text{with probability } \eta \\ 1 & \text{with probability } (1 - \eta), \end{cases} \quad (11)$$

$$h_{i,\pi_{j+1}} = \begin{cases} 0 & \text{with probability } \eta \\ 1 & \text{with probability } (1 - \eta), \end{cases} \quad (12)$$

and for $k = 1, \ldots, n$, where $k \neq j, j + 1$,

$$h_{i,\pi_k} = \begin{cases} 0 & \text{with probability } (1 - \rho) \\ 1 & \text{with probability } \rho. \end{cases} \quad (13)$$

Hence,

$$\begin{aligned} P(H_i \mid \Pi, Y, B_{i,j}) = &(1 - \eta)^{(h_{i,\pi_j} + h_{i,\pi_{j+1}})} \cdot \eta^{(2 - h_{i,\pi_j} - h_{i,\pi_{j+1}})} \\ &\cdot \rho^{(p_i - h_{i,\pi_j} - h_{i,\pi_{j+1}})} \cdot (1 - \rho)^{(n-2) - (p_i - h_{i,\pi_j} - h_{i,\pi_{j+1}})}. \quad (14) \end{aligned}$$

Finally, for a given probe ordering Π and interprobe spacing vector Y, event $N_{i,j}$ implies that all the elements of row H_i should be equal to 0. That is, for $k = 1, \ldots, n$,

$$h_{i,\pi_k} = \begin{cases} 0 & \text{with probability } (1 - \rho) \\ 1 & \text{with probability } \rho. \end{cases} \quad (15)$$

Hence,

$$P(H_i \mid \Pi, Y, N_{i,j}) = \rho^{p_i} \cdot (1 - \rho)^{(n - p_i)}. \quad (16)$$

From Equations 4–16,

$$\begin{aligned} P(H_i \mid \Pi, Y) = &\sum_{j=1}^{n} \left[(1 - \eta)^{h_{i,\pi_j}} \cdot \eta^{(1 - h_{i,\pi_j})} \cdot \rho^{(p_i - h_{i,\pi_j})} \cdot (1 - \rho)^{(n-1) - (p_i - h_{i,\pi_j})} \right. \\ &\left. \cdot \frac{\min(Y_j, M) + \min(Y_{j+1}, M)}{N - M} \right] \\ &+ \sum_{j=1}^{n-1} \left[(1 - \eta)^{(h_{i,\pi_j} + h_{i,\pi_{j+1}})} \cdot \eta^{(2 - h_{i,\pi_j} - h_{i,\pi_{j+1}})} \cdot \rho^{(p_i - h_{i,\pi_j} - h_{i,\pi_{j+1}})} \right. \\ &\left. \cdot \frac{M - \min(Y_{j+1}, M)}{N - M} \right] \\ &+ \sum_{j=0}^{n} \left[\rho^{p_i} \cdot (1 - \rho)^{(n - p_i)} \cdot \frac{Y_{j+1} - \min(Y_{j+1}, M)}{N - M} \right]. \quad (17) \end{aligned}$$

We assume that the clones $\in \mathcal{C}$ are independently distributed along the chromosome; *i.e.*, each row of H is independent of the other rows. Hence

$$P(H \mid \Pi, Y) = \prod_{i=1}^{k} P(H_i \mid \Pi, Y). \quad (18)$$

From Equations 17 and 18,

$$P(H \mid \Pi, Y) = \prod_{i=1}^{k} C_i \left\{ R_i - \sum_{j=1}^{n+1} (a_{i,\pi_j} - 1)(a_{i,\pi_{j-1}} - 1) \min(Y_j, M) \right\}, \quad (19)$$

where

$$a_{i,j} = \begin{cases} \dfrac{\eta}{(1 - \rho)} & \text{if } j_{i,j} = 0 \text{ and } j = 1, \ldots, n \\ \dfrac{(1 - \eta)}{\rho} & \text{if } h_{i,j} = 1 \text{ and } j = 1, \ldots, n \\ 0 & \text{otherwise,} \end{cases} \quad (20)$$

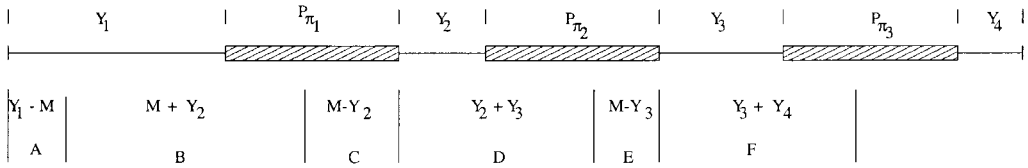

FIGURE 4.—Interprobe spacings: Case 3.

$$C_i = \frac{\rho^{p_i}(1-\rho)^{(n-p_i)}}{N-M}, \tag{21}$$

and

$$R_i = N - nM + M\sum_{j=1}^{(n-1)} a_{i,\pi_j} a_{i,\pi_{j+1}}. \tag{22}$$

The goal, therefore, is to determine Π and Y that maximize $P(H \mid \Pi, Y)$ as given in Equation 19, that is, determine $(\hat{\Pi}, \hat{Y})$, where

$$(\hat{\Pi}, \hat{Y}) = \arg\max_{(\Pi, Y)} P(H \mid \Pi, Y). \tag{23}$$

Alternatively, consider the negative log-likelihood function $f(\Pi, Y)$ given by

$$f(\Pi, Y) = -\ln P(H \mid \Pi, Y)$$
$$= C - \sum_{i=1}^{k} \ln\left\{R_i = \sum_{j=1}^{n+1} (a_{i,\pi_j} - 1)(a_{i,\pi_{j-1}} - 1)\min(Y_j, M)\right\}, \tag{24}$$

where C is a constant given by

$$C = k\ln(N-M) - P\ln\frac{\rho}{(1-\rho)} - nk\ln(1-\rho) \tag{25}$$

and $\pi_0 = \pi_{n+1} = 0$. Since $\ln x$ is a monotonically increasing function of x for all $x > 0$, it follows that

$$(\hat{\Pi}, \hat{Y}) = \arg\max_{(\Pi, Y)} P(H \mid \Pi, Y) = \arg\min_{(\Pi, Y)} f(\Pi, Y). \tag{26}$$

Computation of the ML estimate: Computing the values of $\hat{\Pi}$ and $\hat{Y}$ (Equation 26) involves a two-stage procedure.

Stage 1: First determine the optimal spacing $\hat{Y}_\Pi$ for a given probe ordering Π; *i.e.*, determine $\hat{Y}_\Pi = (\hat{Y}_1, \ldots, \hat{Y}_n)$ such that for a given Π,

$$f(\Pi, \hat{Y}_\Pi) = \min_Y f(\Pi, Y) = \min_Y f_\Pi(Y). \tag{27}$$

Here the minimum is taken over all feasible solutions Y that satisfy the constraints $Y_i \geq 0$; $i = 1, \ldots, n$ and $\Sigma_{i=1}^{n} Y_i \leq N - nM$.

Stage 2: Second determine $\hat{\Pi}$ for which

$$f(\hat{\Pi}, \hat{Y}_{\hat{\Pi}}) = \min_\Pi f(\Pi, \hat{Y}_\Pi) = \min_\Pi f_{\hat{Y}_\Pi}(\Pi). \tag{28}$$

Here the minimum is taken over all Π, where Π is a permutation of $\{1, \ldots, n\}$. The resulting values of $\hat{\Pi}$ and $\hat{Y}_{\hat{\Pi}}$ are termed the ML estimates (MLEs) of the true probe ordering and the interprobe spacings, respectively.

Computation of $\hat{Y}_\Pi$*:* A region $\mathcal{D} \subseteq \mathcal{R}^n$ is deemed to be convex if for any pair of points $p, q \in \mathcal{D}$, all points along the line segment $\alpha p + (1-\alpha)\, q \in \mathcal{D}$, where $0 \leq \alpha \leq 1$. A function h: $\mathcal{D} \rightarrow \mathcal{R}$ defined on a convex set $\mathcal{D}$ is deemed convex if for all points $\alpha p + (1-\alpha)\, q \in \mathcal{D}$, where $0 \leq \alpha \leq 1$, $h(\alpha p + (1-\alpha) q) \leq \alpha h(p) + (1-\alpha) h(q)$. Alternatively, if $(d^2/d\alpha^2) h \geq 0$ along the line segment $\alpha p + (1-\alpha) q$, then function h can be shown to satisfy the above condition for convexity (SHETE 1998). Furthermore, a region $\mathcal{D} \subseteq \mathcal{F}$ is considered *good* if for all $Y \in \mathcal{D}$, $Y_i \neq M$, $1 \leq i \leq n+1$. The significance of a good region is that $f_\Pi(Y)$ is differentiable within it.

The objective function $f_\Pi(Y)$ (Equation 27) can be expressed as

$$f_\Pi(Y) = C - \sum_{i=1}^{k} \ln f_i(Y), \tag{29}$$

where $f_i(Y) = R_i - \Sigma_{j=1}^{n+1} (a_{i,\pi_j} - 1)(a_{i,\pi_{j-1}} - 1) Y_j$. Consider a good convex region $\mathcal{D} \subseteq \mathcal{F}$, where $Y_j \neq M$ for $1 \leq j \leq n$. Consider all points $Y = P + sV$ for $s > 0$, which lie on a ray originating at a given point $P \in \mathcal{C}$ in the direction V. In $\mathcal{C}$ the derivative of f along the ray is given by

$$\frac{d}{ds} f_\Pi(Y) = -\sum_{i=1}^{k} \frac{1}{f_i(Y)} \frac{d}{ds} f_i(Y), \tag{30}$$

where

$$\frac{d}{ds} f_i(Y) = \sum_{j=1}^{n} V_j(-(a_{i,\pi_j} - 1)\ (a_{i,\pi_{j-1}} - 1) \cdot I(Y_j) - (\alpha_{i,\pi_n} - 1) I(Y_{n+1})) \tag{31}$$

and $I(x)$ is a unit step function defined as

$$I(x) = \begin{cases} 1 & \text{if } x < M, \\ 0 & \text{if } x > M, \\ \text{undefined} & \text{if } x = M. \end{cases} \tag{32}$$

Using the fact that $(d^2/ds^2) f_i(Y) = 0$ along the ray, it can be shown that

$$\frac{d^2}{ds^2} f_\Pi(Y) = \sum_{i=1}^{k} \left(\frac{1}{f_i(Y)} \frac{d}{ds} f_i(Y)\right)^2 \geq 0. \tag{33}$$

This implies that $f_\Pi(Y)$ is convex in every good convex region $\mathcal{D}$ and therefore possesses a unique local minimum that is also a global minimum. Consequently this

minimum can be reached using continuous local search-based techniques such as gradient descent (*i.e.*, steepest descent) or conjugate gradient descent (Dorny 1980; Kincaid and Cheney 1991).

Consider the four disjoint subregions $\mathcal{F}_{+1,+1}$, $\mathcal{F}_{+1,-1}$, $\mathcal{F}_{-1,+1}$, and $\mathcal{F}_{-1,-1}$ within $\mathcal{F}$, where

$$\mathcal{F}_{a,b} \triangleq \{Y \in \mathcal{F}: aY_1 \le aM;\ Y_i \le M,\ 2 \le i \le N;\ bY_{n+1} \le bM\}. \tag{34}$$

Each of these regions is convex since they result from the intersection of half spaces. Also, since the derivative of $f_\Pi(Y)$ is defined in the interior of each subregion, each subregion is good. Note that we can define the derivative on the boundary of each subregion $\mathcal{F}_{a,b}$, $a, b \in \{-1, +1\}$, on the basis of the direction in which the boundary point is approached. Thus by selecting a starting point in each of the subregions (or as many subregions as possible without violating any feasibility constraints), one can compute a local minimum for $f_\Pi(Y)$ in each of the subregions and select the minimum of these local minima to be the global minimum of $f_\Pi(Y)$ (Kececioglu *et al.* 2000).

The local minimum of $f_\Pi(Y)$ in each of the aforementioned four disjoint subregions within $\mathcal{F}$ can be reached using continuous local search-based methods such as the steepest descent technique or the conjugate gradient descent technique (Dorny 1980; Kincaid and Cheney 1991). The steepest descent technique is a simple iterative procedure that consists of three steps: (i) Determine the initial value of Y,(ii) compute the downhill gradient at Y, and (iii) update the current value of Y using the computed value of the downhill gradient. Steps (ii) and (iii) are repeated until the gradient vanishes. The point at which the gradient vanishes is considered to be the desired local minimum. In practice, the steepest descent procedure is halted when the magnitude of the gradient is less than a prespecified threshold.

The initial value of $Y = (Y_1, \ldots, Y_n)$ can be determined in one of several ways; the simplest solution is to assign $(N - nM)/(n + 1)$ to each of Y_i's, *i.e.*, distribute the value of $N - nM$ equally among the $(n + 1)$ Y_i's. Having obtained an initial value for $\hat{Y}$, the gradient of the negative log-likelihood function is computed. A function decreases most rapidly in the direction of the local negative (*i.e.*, downhill) gradient. The local downhill gradient is given by

$$\begin{aligned} -\nabla f(\Pi, \hat{Y}) &= -\left(\frac{\partial f(\Pi, Y)}{\partial Y_1}, \ldots, \frac{\partial f(\Pi, Y)}{\partial Y_n}\right)\Big|_{Y=\hat{Y}} \\ &= (U_1, \ldots, U_n)|_{Y=\hat{Y}}, \end{aligned} \tag{35}$$

where

$$U_l = \sum_{i=1}^{k} \frac{-(a_{i,\pi_l} - 1)(a_{i,\pi_{l-1}} - 1)I(Y_l) - ((a_{i,\pi_n} - 1)I(Y_{n+1}))}{R_i - \Sigma_{j=1}^{n+1}(a_{i,\pi_j} - 1)(a_{i,\pi_{j-1}} - 1)\min(Y_j, M)}, \tag{36}$$

and

$$I(x) = \begin{cases} 1 & \text{if } x \le M \\ 0 & \text{otherwise.} \end{cases} \tag{37}$$

The current value of $\hat{Y} = \hat{Y}_{\text{old}}$ is updated by moving along the downhill gradient direction $U = -\nabla f(\Pi, \hat{Y})|_{Y=\hat{Y}_{\text{old}}}$. The new value of $\hat{Y} = \hat{Y}_{\text{new}}$ is given by

$$\hat{Y}_{\text{new}} = \hat{Y}_{\text{old}} + sU. \tag{38}$$

The problem, therefore, is to find an optimal value of s, say s^*, such that

$$f(\Pi, \hat{Y} + s^*U) = \min_s f(\Pi, \hat{Y} + sU). \tag{39}$$

Here,

$$f(\Pi, \hat{Y} + sU) = C - \sum_{i=1}^{k} \ln\left\{R_i - \sum_{j=1}^{n+1}(a_{i,\pi_j} - 1)(a_{i,\pi_{j-1}} - 1)\min(\hat{Y}_j + sU_j, M)\right\}, \tag{40}$$

where $\hat{Y}_{n+1} = N - nM - \Sigma_{i=1}^{n} \hat{Y}_i$.

Having obtained the value of s^*, then the new interprobe spacings are given by

$$\hat{Y}_{\text{new}} = \hat{Y}_{\text{old}} + s^*U. \tag{41}$$

To determine an optimal value of $s = s^*$, consider

$$\frac{\partial f(\Pi, \hat{Y} + sU)}{\partial s} = \sum_{i=1}^{k} \frac{\Sigma_{j=1}^{n+1}(a_{i,\pi_j} - 1)(a_{i,\pi_{j-1}} - 1)U_j I(\hat{Y}_j + sU_j)}{R_i - \Sigma_{j=1}^{n+1}(a_{i,\pi_j} - 1)(a_{i,\pi_{j-1}} - 1)\min(\hat{Y}_j + sU_j, M)}, \tag{42}$$

where $U_{n+1} = -\Sigma_{i=1}^{n} U_i$. The convexity of $f_\Pi(Y)$ implies that the local optimum for s is also a global optimum. Also note that we have the following boundary conditions: (i) $\hat{Y}_j + sU_j \ge 0$, for $j = 1, \ldots, n$; *i.e.*, all the spacings are nonnegative; and (ii) $\Sigma_{j=1}^{n}(\hat{Y}_j + sU_j) \le N - nM$, which is a constraint imposed by the length of the chromosome and the length of each clone.

The above constraints impose bounds on s given by

$$0 \le s \le \min\left\{\min_{j \in (1,\ldots,n+1):Uj<0}\left\{\frac{-\hat{Y}_j}{U_j}\right\}, \min_{j \in (1,\ldots,n+1):Uj<0}\left\{\frac{M - \hat{Y}_j}{U_j}\right\}\right\}. \tag{43}$$

Given the upper and lower bounds on the values of s and the convexity of $f(\Pi, \hat{Y} + sU)$ with respect to s, the bisection method (Kincaid and Cheney 1991) can be used to find the optimal value of s. The number of iterations of the bisection method required to localize the minimum within a given tolerance ϵ is given by $n = \log_2[(b - a)/\epsilon]$.

The gradient computation and the solution update steps of the steepest descent method are continued until the gradient vector attains a magnitude less than a predefined threshold. During this iterative process, it may happen that the current value of Y is on one or more of the boundaries of the feasible region in which the solution is located. These boundaries are defined by

the constraints on the Y_i's for $i = 1, \ldots, n$ as discussed earlier. The fact that the current value of Y lies on one or more of the boundaries of the feasible region and downhill gradient vector is determined to point outside the feasible region would normally force the value of s to be equal to zero and hence stop the iterative procedure even though the gradient vector has not vanished.

The above difficulty can be overcome by projecting the downhill gradient vector U onto the feasible region. Note that all the boundary conditions on the Y_i's for $i = 1, \ldots, n$ are hyperplanes. Suppose that U is directed outside the feasible region, and we have k hyperplanes (corresponding to k boundary conditions) that force s to be equal to zero. In this situation one needs to project U onto the intersection of these hyperplanes. Let $\vec{N}_1, \ldots, \vec{N}_k$ be the normal vectors to these hyperplanes. If the boundary conditions are not redundant, then $\{\vec{N}_1, \ldots, \vec{N}_k\}$ constitutes a linearly independent set of vectors. The set $\{\vec{N}_1, \ldots, \vec{N}_k\}$ can be transformed into a mutually orthonormal set of vectors given by $\{\vec{N}_1, \ldots, \vec{N}_k'\}$ using the Gram-Schmidt orthonormalization procedure (Dorny 1980). The projected vector U_{proj} is then given by:

$$U_{\text{proj}} = U - (U \cdot \vec{N}_1')\vec{N}_1' - \ldots - (U \cdot \vec{N}_k')\vec{N}_k'. \tag{44}$$

The minimization procedure then proceeds along U_{proj} instead of U. In the limiting case when $k = n$, the minimization procedure has reached an extremal vertex of the admissible region and $U_{\text{proj}} = 0$. In this case, the extremal vertex is the desired minimum within the admissible region. Thus, the minimization procedure is halted when U vanishes or when an extremal vertex is reached (*i.e.*, U_{proj} vanishes) depending on which situation is encountered first.

Computation of $\hat{\Pi}$*:* Determining the optimal clone ordering $\hat{\Pi}$ entails a combinatorial search through the discrete space of all possible permutations of $\{1, \ldots, n\}$. The problem of coming up with such an optimal ordering is isomorphic to the classical nondeterministic polynomial-complete *optimal linear arrangement* (OLA) problem for which no polynomial-time algorithm for determining the optimal solution is known (Garey and Johnson 1979). Heuristic search algorithms that are capable of arriving at near-optimal solutions in polynomial time on average are therefore desirable. However, deterministic hill-climbing (*i.e.*, "local" or "greedy") search algorithms have a tendency to get trapped in a *local* optimum that may be far from a desirable *global* optimum. An attractive alternative is to use a stochastic hill-climbing search algorithm such as simulated annealing (SA; Kirkpatrick *et al.* 1983; Geman and Geman 1984) or microcanonical annealing (MCA; Creutz 1983), both of which are known to be robust in the presence of local optima in the solution space.

A single iteration of a stochastic hill-climbing search algorithm consists of three phases: (i) perturb, (ii) evaluate, and (iii) decide. In the perturb phase, the current solution $\mathbf{x}_i$ to a multivariate objective function $E(\mathbf{x})$, which is to be minimized, is systematically perturbed to yield another candidate solution $\mathbf{x}_j$. Here, the probe ordering is systematically perturbed by reversing the ordering within a block of probes where the endpoints of the block are chosen at random. In the evaluate phase, $E(\mathbf{x}_j)$ is computed. Here, the function $f(\Pi, \hat{Y}_\Pi)$ (Equation 27) is computed. In the decide phase, $\mathbf{x}_j$ is accepted and replaces $\mathbf{x}_i$ *probabilistically*, using a stochastic decision function. The stochastic decision function is *annealed* in a manner such that the search process resembles a random search in the earlier stages and a greedy local search or a deterministic hill-climbing search in the latter stages. The major difference between simulated annealing and microcanonical annealing arises from the difference in the stochastic decision function used in the decision phase. Their common feature is that, starting from an initial solution, they generate, in the limit, an ergodic Markov chain of solution states that asymptotically converges to a stationary Boltzmann distribution (Aarts and Korst 1989). The Boltzmann distribution asymptotically converges to a globally optimal solution when subjected to the annealing process (Geman and Geman 1984).

Simulated annealing: In the decide phase of SA, the new candidate solution $\mathbf{x}_j$ is accepted with probability p, which is computed using the Metropolis function (Metropolis *et al.* 1953)

$$p = \begin{cases} 1 & \text{if } E(\mathbf{x}_j) < E(\mathbf{x}_i) \\ \exp\left(-\dfrac{E(\mathbf{x}_j) - E(\mathbf{x}_i)}{T}\right) & \text{if } E(\mathbf{x}_j) \geq E(\mathbf{x}_i) \end{cases} \tag{45}$$

or, using the Boltzmann function $B(T)$,

$$p = \mathrm{B}(T) = \frac{1}{1 + \exp([E(\mathbf{x}_j) - E(\mathbf{x}_i)]/T)} \tag{46}$$

(Aarts and Korst 1989) at a given value of temperature T, whereas $\mathbf{x}_i$ is retained with probability $(1 - p)$.

The Metropolis function and the Boltzmann function give SA the capability of climbing out of local minima. Several iterations of SA are carried out for a given value of T and are referred to as an *annealing step*. The value of T is systematically reduced using an annealing function. As can be seen from Equations 45 and 46, at sufficiently high temperatures SA resembles a completely random search, whereas at lower temperatures it acquires the characteristics of a deterministic hill-climbing (*i.e.*, local or greedy) search.

SA generates an asymptotically ergodic (and hence stationary) Markov chain of solution states at a given temperature using Equations 45 and 46. Logarithmic annealing schedules of the form $T_k = R/\log k$ for some value of $R > 0$ have been shown to be asymptotically good (Geman and Geman 1984); *i.e.*, they ensure asymptotic convergence to a global minimum with unit probability in the limit $k \rightarrow \infty$ (Geman and Geman 1984). The convergence criterion used here was the fact that

the value of the objective function had not changed for the past k annealing steps. A geometric annealing schedule of the form $T_{k+1} = \alpha T_k$ was used where $\alpha = 0.9$. Although the geometric annealing schedule does not strictly ensure asymptotic convergence to a global optimum as does the logarithmic annealing schedule, it is much faster and has been found to yield very good solutions in practice (Romeo and Sangiovanni-Vincentelli 1991).

Microcanonical annealing: MCA models a physical system whose total energy, *i.e.*, the sum of kinetic energy and potential energy, is always conserved. The potential energy of the system is the multivariate objective function $E(\mathbf{x})$ to be minimized, whereas the kinetic energy $E_k > 0$ is represented by a demon or a collection of demons. In the latter case, the total kinetic energy is the sum of all the demon energies. The demon energy (or energies) serve(s) to provide the system with an extra degree (or degrees) of freedom thus enabling MCA to escape from local minima.

In the decide phase of MCA, if $E(\mathbf{x}_j) < E(\mathbf{x}_i)$, then $\mathbf{x}_j$ is accepted as the new solution. If $E(\mathbf{x}_j) \geq E(\mathbf{x}_i)$, then $\mathbf{x}_j$ is accepted as the new solution only if $E_k \geq E(\mathbf{x}_j) - E(\mathbf{x}_i)$. If $E(\mathbf{x}_j) \geq E(\mathbf{x}_i)$ and $E_k < E(\mathbf{x}_j) - E(\mathbf{x}_i)$ then the current solution $\mathbf{x}_i$ is retained. In the event that $\mathbf{x}_j$ is accepted as the new solution, the kinetic energy demon is updated to $E_k^{n+1} = E_k^n + [E(\mathbf{x}_i) - E(\mathbf{x}_j)]$ to ensure the conservation of the total energy. The kinetic energy parameter E_k is annealed in a manner similar to the temperature parameter T in SA. MCA can also be shown to converge to a global minimum with unit probability given a logarithmic annealing schedule (Bhanot *et al.* 1984).

In the context of probe ordering, a kinetic energy demon is assigned to each distinct pair of probes. A square symmetric matrix E_k of size $n \times n$, where n is the number of probes, is used to store the demon energy for each probe pair. Thus, an entry $E_k(i, j)$ refers to the kinetic energy of the demon associated with the ith and jth probe. As in the case of SA, the convergence criterion used was the fact that the value of the objective function had not changed for the past k annealing steps. A geometric annealing schedule of the form $E_k^{m+1}(i, j) = \alpha E_k^m(i, j)$ was used where $\alpha = 0.9$.

In spite of the robustness of SA and MCA to the presence of local minima in the solution landscape, the annealing schedule needed for asymptotic convergence is computationally intensive. This provides the motivation for the parallel computation of the maximum-likelihood estimator.

PARALLEL COMPUTATION OF THE MAXIMUM-LIKELIHOOD ESTIMATOR

Computation of the maximum-likelihood estimator entails two levels of parallelism corresponding to the two stages of optimization discussed in the previous section:

Level 1: Parallel computation of the optimal interprobe spacing $\hat{Y}_\Pi$ for a given probe ordering Π (Equation 27). This entails parallelization of the gradient descent search procedure for constrained optimization in the *continuous* domain.

Level 2: Parallel computation of the optimal probe ordering (Equation 28). This entails parallelization of the stochastic hill-climbing search procedure (SA or MCA) for optimization in the *discrete* domain.

Both levels of parallel computation were implemented on PVM (Sunderam 1990), which is a software environment designed to exploit parallel/distributed computing across a variety of computing platforms. PVM is based on a distributed-memory message-passing paradigm of parallel computing. The interested reader is referred to Geist *et al.* (1994) for a more detailed description of PVM.

Parallel stochastic hill-climbing search: Parallelization of annealing algorithms has been attempted by several researchers especially in the area of computer-aided design, image processing, and operations research (Greening 1990; Azencott 1992). Parallelization strategies for the SA and MCA algorithms can be categorized as follows:

A. Functional parallelism within a move where the task of evaluating each move is decomposed into subtasks that are performed in parallel by multiple processors (Wong and Fiebrich 1987).
B. Control parallelism with multiple active iterations with processors engaged in *speculative* computation (Witte *et al.* 1991).
C. Control parallelism with parallel Markov chain generation using a systolic algorithm (Aarts *et al.* 1986; Greening 1990; Kim and Kim 1990; Azencott 1992).
D. Control parallelism with multiple searches of the solution space where the searches could be interacting, in which case the processors exchange data periodically, or noninteracting in which case the processors proceed independently of each other (Azencott 1992; Lee 1995).
E. Data parallelism with the state variables in a multivariate solution vector distributed among the individual processors in the multiprocessor architecture accompanied by either (1) parallel evaluation of multiple moves with acceptance of a single move (Casotto *et al.* 1987), or (2) parallel evaluation of multiple moves with acceptance of multiple noninteracting moves (Jayaraman and Rutenbar 1987), or parallel evaluation and acceptance of multiple moves (Banerjee *et al.* 1990).

The authors' earlier work in chromosome reconstruction involved ordering clones, *i.e.*, rows of the hybridization matrix H as shown in Table 1 for physical map generation (Cuticchia *et al.* 1992, 1993). The clones were ordered on the basis of minimization of the total

pairwise Hamming distance between the binary hybridization signatures of successive clones in a given permutation. This clone ordering problem was also shown to be isomorphic to the optimal linear arrangement problem (Garey and Johnson 1979), and SA and MCA were used to arrive at the optimal clone ordering, which was represented by a global minimum of the total pairwise Hamming distance objective function.The authors' efforts at parallelizing SA and MCA in the context of the aforementioned clone ordering problem showed control parallelism based on multiple searches to be the most promising for implementation on a distributed memory multiprocessor platform such as PVM (Bhandarkar *et al.* 1996a,b, 1998; Bhandarkar 1997; Bhandarkar and Machaka 1997). Since a candidate solution in the serial SA or MCA algorithm can be considered to be an element of an asymptotically ergodic first-order Markov chain of solution states, two models of parallel SA (PSA) and parallel MCA (PMCA) algorithms were formulated and implemented based on the distribution of the Markov chain of solution states on a workstation cluster running PVM. These models incorporate the parallelization strategies discussed under item (D) above and are described below:

The noninteracting local Markov chain (NILM) PSA and PMCA algorithms.
The periodically interacting local Markov chain (PILM) PSA and PMCA algorithms.

In the NILM PSA and NILM PMCA algorithms, each processor within the PVM system runs an independent version of the serial SA or MCA algorithm. In essence, there are as many Markov chains of solution states as there are physical processors within the PVM system. Each Markov chain is local to a given processor and at any instant each processor maintains a candidate solution that is an element of its local Markov chain of solution states. The serial SA and MCA algorithms run asynchronously on each processor; *i.e.*, at each temperature value or kinetic energy value each processor iterates through the perturb-evaluate-accept cycle concurrently (but asynchronously) with all the other processors.

The perturbation function uses a parallel random number generator to generate the Markov chains of solution states. By assigning a distinct seed to each processor at the start of execution, it is ensured that each processor contains a Markov chain of solution states that is independent from those in most or all of the other processors. The evaluation function and the decision function are executed concurrently on the solution state within each processor. On termination of the annealing processes on all the processors, the best solution is selected from among all the solutions available on the individual processors. The NILM model is essentially that of multiple independent (*i.e.*, noninteracting) searches as described in item (D) above.

The PILM PSA and PILM PMCA algorithms are similar to their NILM counterparts except for one major difference. Just before the parameter T or E_k is updated using the annealing function, the best candidate solution from among those in all the processors is selected and duplicated on all the other processors. The goal of this synchronization procedure is to focus the search in the more promising regions of the solution space. This suggests that the PILM PSA or PILM PMCA algorithm should be potentially superior to its NILM counterpart. It should be noted that in the case of all four algorithms, PILM PSA, PILM PMCA, NILM PSA, and NILM PMCA, a single Markov chain of solution states is generated entirely within a single processor. The PILM model is essentially that of multiple periodically interacting searches as described in item (D) above.

In the case of all the four algorithms, NILM PSA, NILM PMCA, PILM PSA, and PILM PMCA, a master process is used as the overall controlling process. The master process runs on one of the processors within the PVM system. The master process spawns child processes on each of the other processors within the PVM system, broadcasts the data subsets needed by each child process, collects the final results from each of the child processes, and terminates the child processes. The master process, in addition to the above-mentioned functions for task initiation, task coordination, and task termination, also runs its own version of the SA or MCA algorithm just as does any of its child processes.

In the case of the PILM PSA and the PILM PMCA algorithms, before the parameter T or E_k is updated, the master process collects the results from each child process along with its own result, broadcasts the best result to all the child processes, and also replaces its own result with the best result. The master process updates its temperature or kinetic energy value using the annealing schedule and proceeds with its local version of the SA or MCA algorithm. On convergence, the master process collects the final results from each of the child processes along with its own, selects the best result as the final solution, and terminates the child processes.

Each of the child processes in the PILM PSA and the PILM PMCA algorithms receives the initial parameters from the master process and runs its local version of the SA or MCA algorithm. At the end of each annealing step, each child process conveys its result to the master process, receives the best result thus far from the master process, and replaces its result with the best result thus far before proceeding with the next annealing step. On convergence, each child process conveys its result to the master process. The master process and child process for the PILM PSA and the PILM PMCA algorithms are depicted in Figures 5 and 6, respectively.

The master and child processes for the NILM PSA and NILM PMCA algorithms are similar to those of their PILM counterparts except for the absence of the process coordination phase (Figures 5 and 6) in the former. Note that it is during the process coordination phase

```
master()
beginmaster

Phase 1: Initial Setup
    (a) Spawn Child Processes;
    (b) Read Input: probe and clone data, hybidization matrix H, initial probe ordering and
    inter-probe spacings and initial seeds for the parallel random number generator;
    (c) Broadcast the input to all spawned child processes;

Phase 2: Annealing and Process Coordination
    if (PSA) T = T_max;
    if (PMCA) E_k = E_max;
    while (not Converged)
    beginwhile

      Annealing Step
          while (current annealing step)
          beginwhile
            (a) Perturb Phase: same as serial SA or MCA;
            (b) Evaluate Phase: same as serial SA or MCA;
            (c) Decide Phase:
                if (PSA) SA decision criterion;
                if (PMCA) MCA decision criterion;
            (d) Evaluate annealing step termination criterion;
          endwhile
      Process Coordination
          Receive probe ordering and inter-probe spacing from each child process;
          Select best probe ordering and inter-probe spacing;
          Send best probe ordering and inter-probe spacing to each child process;
          if PSA: Update temperature T = A(T);
          if PMCA: Update kinetic energy E_k = A(E_k);

    Evaluate convergence criterion;
    endwhile

Phase 3: Output Result
    Receive probe ordering and corresponding inter-probe spacing from each child process;
    Select best probe ordering and corresponding inter-probe spacing;
    Output best probe ordering and corresponding inter-probe spacing;

endmaster
```

FIGURE 5.—The master process for the PILM PSA/PMCA algorithm.

at the end of each annealing step that the master process and the child processes interact in the PILM PSA and PILM PMCA algorithms.

Parallel gradient descent: Deterministic optimization techniques such as steepest descent and conjugate gradient descent are generally used for unconstrained optimization in the continuous domain (POLAK 1997). However, the steepest descent procedure in this case needs to be adapted to the fact that the solution space of the interprobe spacings is constrained, since $0 \leq Y_i \leq M$ for $i = 1, \ldots, n$. One of the well-known problems with the steepest descent method is that it takes several small steps while descending a valley in the solution landscape, which usually causes it to be much slower compared to other techniques in its class (POLAK 1997). The conjugate gradient descent (CGD) procedure, on the other hand, is known to be one of the fastest in the class of gradient descent-based optimization methods (HESTENES 1980; HESTENES and STIEFEL 1952).

The conjugate gradient descent procedure is very similar to the steepest descent procedure with the only difference that different directions are followed while minimizing the objective function. Instead of consistently following the local downhill gradient (*i.e.*, the direction of steepest descent), a set of n mutually orthonormal (*i.e.*, conjugate) direction vectors are generated from the downhill gradient vector where n is the dimensionality of the solution space. The orthonormality condition ensures that minimization along any given direction vector does not jeopardize the minimization along another direction vector within this set. Unlike the steepest descent algorithm, the conjugate gradient descent algorithm guarantees convergence to a local mini-

child()
beginchild

Phase 1: *Initial Setup:*
Receive Input from Master Process: probe and clone data, hybidization matrix H, initial probe ordering and inter-probe spacing and initial seeds for the parallel random number generator;

Phase 2: *Annealing and Process Coordination*
if (PSA) $T = T_{max}$;
if (PMCA) $E_k = E_{max}$;
while (not Converged)
beginwhile

Annealing Step
while (current annealing step)
beginwhile
(a) Perturb Phase: same as serial SA or MCA;
(b) Evaluate Phase: same as serial SA or MCA;
(c) Decide Phase:
if (PSA) SA decision criterion;
if (PMCA) MCA decision criterion;
(d) Evaluate annealing step termination criterion;
endwhile

Process Coordination
Send probe ordering and inter-probe spacing to master process;
Receive best probe ordering and inter-probe spacing from master process;
if PSA: Update temperature $T = A(T)$;
if PMCA: Update kinetic energy $E_k = A(E_k)$;

Evaluate convergence criterion;
endwhile

Phase 3: *Send Final Result*
Send probe ordering and corresponding inter-probe spacing to master process;

endchild

FIGURE 6.—The child process for the PILM PSA/PMCA algorithm.

mum within n steps. For this reason, the conjugate gradient descent algorithm was chosen instead of the steepest descent algorithm in the current implementation. The serial conjugate gradient descent algorithm is depicted in Figure 7. The Hessian matrix is usually required to generate a set of conjugate vectors to minimize a particular objective function (DORNY 1980; KINCAID and CHENEY 1991). However, the conjugate gradient descent algorithm presented in PRESS *et al.* (1988) describes a method for generating conjugate direction vectors without the need for evaluating the Hessian matrix. The conjugate gradient descent algorithm depicted in Figure 7 is an adaptation of the one presented in PRESS *et al.* (1988).

Due to its inherently sequential nature, data parallelism was deemed to be the appropriate parallelization scheme for the conjugate gradient descent algorithm. The Y and G vectors are distributed among the different processors constituting the virtual machine. Each processor performs the required operations on its local Y_{loc} and G_{loc} subvectors concurrently with the other processors. Here, $|Y_{\text{loc}}| = |Y|/N_{\text{proc}}$ and $|G_{\text{loc}}| = |G|/N_{\text{proc}}$, where N_{proc} is the number of processors in the virtual machine. This entails some interprocessor communication and synchronization overhead since the individual subvectors have to be periodically scattered (*i.e.*, distributed) among the processors and also periodically gathered (*i.e.*, combined) to compute a global value. For example, in the bisection procedure (*i.e.*, procedure for minimizing along the direction G) one needs to evaluate the objective function and compute its one-dimensional derivative with respect to s during each iteration. Both of these operations require gathering of data (*i.e.*, subvectors) within the different processors.

PVM permits one to define a group of processes such that reduction, scattering, and gathering operations can be performed selectively on the processes within the group. The pvm_scatter and pvm_gather primitives were used to scatter and gather the Y and G vectors among the different processors, respectively. It is to be noted, however, that scatter and gather operations in PVM are collective calls and need to be performed by

Conjugate Gradient Descent Algorithm:

Phase 1: *Initialization*

Start with an initial guess of $Y = Y_i$;
Calculate gradient $G = \nabla f(\Pi, Y_i)$;
$G = G_1 = G_2 = -G$;

Phase 2: *Iterative Refinement*

while (1)
beginwhile

Project G ;
if (G has vanished) break;
Bracket the minimum along the direction G;
Minimize along the direction G:
Find the optimal s^* such that $f(\Pi, Y_i + s^*G) = \min_s f(\Pi, Y_i + sG)$;
$Y_{i+1} = Y_i + s^*G$;
$\Delta f = f(\Pi, Y_i) - f(\Pi, Y_{i+1})$;
$Y_i = Y_{i+1}$;
if ($\Delta f < \epsilon$) break;
Calculate gradient $G = \nabla f(\Pi, Y_i)$;
$g_1 = (G + G_2) \cdot G$;
$g_2 = G_2 \cdot G_2$;
$g_3 = g_1/g_2$;
$G_2 = -G$;
$G = G_1 = G_2 + g_3 G_1$;

endwhile

Phase 3: *Output Result*

$Y = Y_i$;

FIGURE 7.—Serial conjugate gradient descent algorithm.

all the processors concurrently. Synchronizing the processors, therefore, is necessary before any such call is initiated, thereby increasing the parallelization overhead. The parallelization scheme for the conjugate gradient descent algorithm follows the master-child model used for the PSA and PMCA algorithms. The master and child processes are depicted in Figures 8 and 9, respectively.

A two-level parallelism approach for computation of the maximum-likelihood estimator: To ensure a scalable implementation, two levels of parallelism were incorporated in the computation of the maximum-likelihood estimator. The finer or lower level of parallelism pertains to the computation of $\hat{Y}$ for a given probe ordering Π using the parallel conjugate gradient descent algorithm for continuous optimization as discussed previously. The coarser or upper level of parallelization pertains to the computation of $\hat{\Pi}$ using the NILM or PILM, PSA or PMCA algorithm for discrete stochastic optimization as discussed previously. The two-level parallelization scheme is depicted in Figure 10.

At the coarser level, the user has a choice of any of the four parallel stochastic hill-climbing algorithms: PILM PSA, NILM PSA, PILM PMCA, or NILM PMCA. The parallelization of the conjugate gradient descent algorithm at the finer level is transparent to the parallel stochastic hill-climbing algorithms at the coarser level. In other words, the communication and control scheme for the parallel stochastic hill-climbing algorithms is independent of that of the parallel conjugate gradient descent algorithm. This enhances the modularity and flexibility of the system. For example, one could use the serial or parallel version of the conjugate gradient descent algorithm or, for that matter, any other serial

master()
beginmaster

Phase 1: *Initial Setup:*
Start with an initial guess of $Y = Y_i$;
Broadcast initialization data to the child processes;
Divide up Y_i vector among children and scatter it;
Let Y_{m_i} be the master's portion of Y_i and Y_{c_i} be the child c's portion of Y_i;
Compute the master's portion of the gradient corresponding to Y_{m_i}: $G_m = \nabla f(\Pi, Y_{m_i})$;
$G_m = G_{m_1} = G_{m_2} = -G_m$;

Phase 2: *Iterative Refinement:*
while (1)
beginwhile
Project G_m;
if (G_m has vanished) *exit-flag* = 1;
Receive the *exit-flag* boolean variables from all the child processes;
If all *exit-flag* variables from master and children are 1 then *exit-flag* = 1; else *exit-flag* = 0;
Broadcast the value of *exit-flag* to the child processes;
If (*exit-flag* = 1), break;
Bracket the minimum along the direction G_m;
Gather the bracketing information from the child processes and the master process in the master process;
Broadcast the bracketing information to the child processes;
Minimize along the direction G: Compute the optimal s^* such that $f(\Pi, Y_i + s^*G) = \min_s f(\Pi, Y_i + sG)$;
Note: *In order to implement the above line minimization procedure, some scatter/gather operations between the master process and child processes are needed.*
Broadcast s^* to the child processes;
$Y_{m_{i+1}} = Y_{m_i} + s^*G_m$;
$Y_{m_i} = Y_{m_{i+1}}$;
Gather the values of $f(\Pi, Y_i)$ and $f(\Pi, Y_{i+1})$ in the master process;
$\Delta f = f(\Pi, Y_i) - f(\Pi, Y_{i+1})$;
if ($\Delta f < \epsilon$) *exit-flag* = 1;
Broadcast the value of *exit-flag* to the child processes;
If *exit-flag* = 1, break;
Calculate gradient $G_m = \nabla f(\Pi, Y_{m_i})$;
$g_1 = (G_m + G_{m_2}) \cdot G_m$;
$g_2 = G_{m_2} \cdot G_{m_2}$;
Gather g_1 and g_2 from the master process and child processes in the master process;
Broadcast g_1 and g_2 to the child processes;
$g_3 = g_1/g_2$;
$G_{m_2} = -G_m$;
$G_m = G_{m_1} = G_{m_2} + g_3G_{m_1}$;
endwhile

Phase 3: *Collect Results:*
Gather Y_i from the master process and child processes in the master process;
$Y = Y_i$;

endmaster

FIGURE 8.—Master process for the parallel conjugate gradient descent algorithm.

or parallel algorithm for continuous deterministic optimization at the finer level without having to make any changes to the parallel stochastic hill-climbing algorithms at the coarser level.

The interaction between the master and child stochastic hill-climbing (*i.e.*, annealing) processes (NILM/PILM PSA/PMCA) is shown with the double-headed arrows between the larger components in Figure 10. A parallel conjugate gradient descent algorithm is embedded within each of the stochastic hill-climbing processes. When the parallel conjugate gradient descent procedure is invoked from within the master or child stochastic hill-climbing process, a new set of child conjugate gradient descent processes is spawned on the available processors, whereas the master conjugate gradient descent process runs on the same processor as the stochastic hill-climbing process (master or child). The master and child conjugate gradient descent processes cooperate to evaluate and minimize the value of the objective function for a specific probe ordering Π. Once the objective function is evaluated, the child conjugate gradient descent processes terminate, and the corresponding processors are available for future computation. The interaction between the master and child conjugate gra-

child()
beginchild

Phase 1: *Initial Setup:*
Receive Y_{c_i} and initialization data from the master process;
Calculate gradient $G_c = \nabla f(\Pi, Y_{c_i})$;
$G_c = G_{c_1} = G_{c_2} = -G_c$;

Phase 2: *Iterative Refinement:*
while (1)
beginwhile
Project G_c ;
if (G_c has vanished) *exit-flag* = 1;
Send *exit-flag* to the master process;
Receive *exit-flag* from the master process;
If *exit-flag* = 1, break;
Bracket the minimum along the direction G_c
Send the local bracketing information to the master process;
Receive the bracketing information to the children;
Minimize along the direction G as mentioned in the master process;
Note: *The above step entails scatter-gather operations between the master process and the child processes*
Receive s^*;
$Y_{c_{i+1}} = Y_{c_i} + s^* G_c$;
$Y_{c_i} = Y_{c_{i+1}}$;
Send the values of $f(\Pi, Y_i)$ and $f(\Pi, Y_{i+1})$ to the master process;
Receive *exit-flag* from the master process;
If *exit-flag* = 1 break;
Calculate gradient $G_c = \nabla f(\Pi, Y_{c_i})$;
$g_1 = (G_c + G_{c_2}) \cdot G_c$;
$g_2 = G_{c_2} \cdot G_{c_2}$;
Send g_1 and g_2 from all the processes to the master;
Receive g_1 and g_2;
$g_3 = g_1/g_2$;
$G_{c_2} = -G_c$;
$G_c = G_{c_1} = G_{c_2} + g_3 G_{c_1}$;
endwhile

Phase 3: *Send results to master process:*
Send Y_{c_i} to the master process;

endchild

FIGURE 9.—Child process for the parallel conjugate gradient descent algorithm.

dient descent processes is depicted by the double-headed arrows within each of the major components in Figure 10.

The two-level parallelism approach can be seen to induce a logical tree-shaped interconnection network on the processors in the PVM system. The first level is the group of processors that run the (master and child) stochastic hill-climbing processes, and the second level is the group of processors that run the child conjugate gradient descent processes. The processors that run the child conjugate gradient descent processes are connected to the processor running the parent stochastic hill-climbing process but are independent of the processors running other stochastic hill-climbing processes.

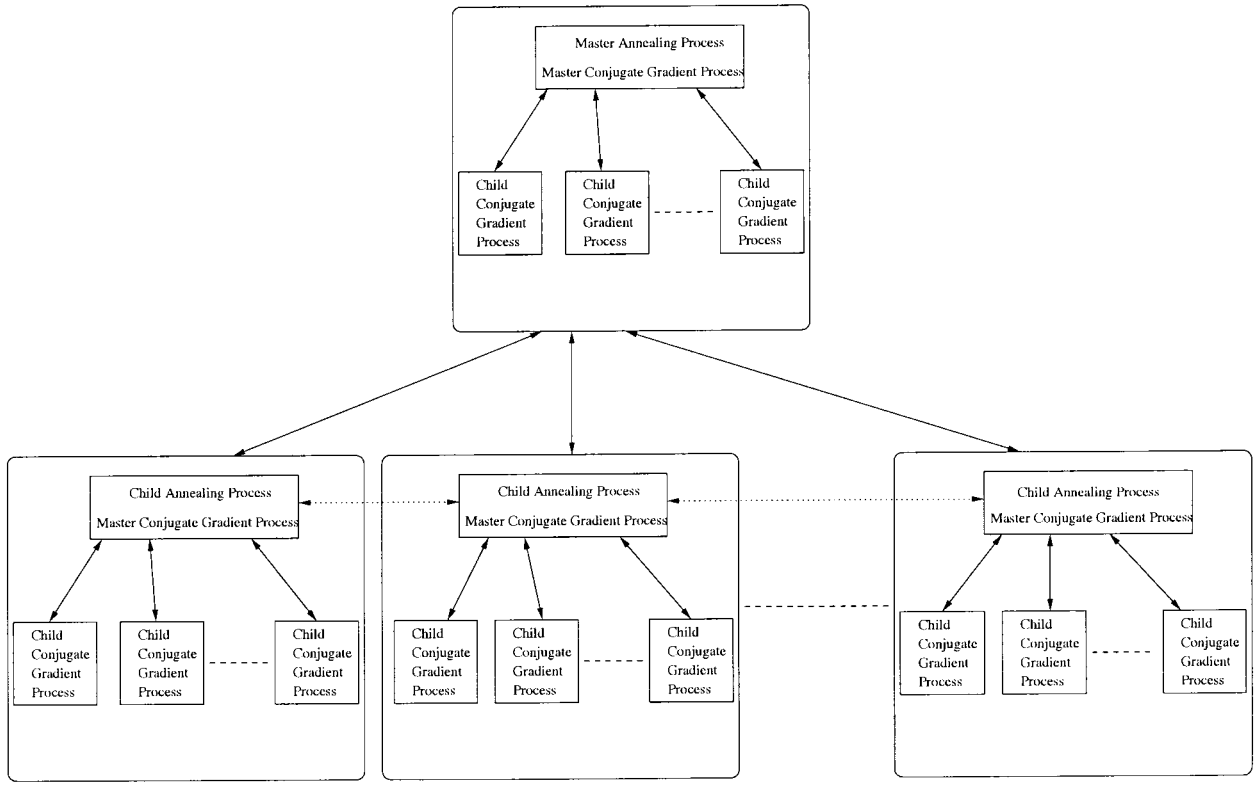

FIGURE 10.—The two-level parallel computation of the maximum-likelihood estimator.

EXPERIMENTAL RESULTS

Experimental results on simulated data: The parallel algorithms were implemented on a dedicated PVM cluster of 200-MHz PentiumPro processors running Solaris-x86 and tested with artificially generated clone-probe hybridization data (SHETE 1998). Two sets of artificial data were used with the specifications outlined in Table 2. The artificial data were generated using a program described in SHETE (1998), which generates clonal data of a given length with the left endpoints of the clones and probes uniformly distributed along the length of an artificial chromosome.

The serial stochastic hill-climbing algorithms (SA and MCA) were implemented with the following parameters: the initial value for the temperature or demon energy was chosen to be 0.5, and the maximum number of iterations D for each annealing step was chosen to be $100 \cdot n$. The current annealing step was terminated when the maximum number of iterations was reached or when the number of successful perturbations equaled $10 \cdot n$ (*i.e.*, 10% of the maximum number of iterations), whichever was encountered first. The temperature or demon energy values were systematically reduced at the end of each annealing step, using a geometric annealing schedule with the annealing factor $\alpha = 0.95$. The algorithm was terminated (and deemed to have reached a global optimum) when the number of successful perturbations in any annealing step equaled 0.

TABLE 2

Specifications of the artificially generated clone-probe hybridization data

Data set	n	k	N	M	ρ (%)	η (%)
Data set 1	10	100	180	15	2	2
Data set 2	30	400	680	20	2	2

In the case of the parallel stochastic hill-climbing algorithms (NILM PSA, PILM PSA, NILM MCA, and PILM MCA) the product of N_{proc} and the maximum number of iterations D performed by a processor in a single annealing step was kept constant, *i.e.*, $D = (100 \cdot n)/N_{proc}$. This ensured that the overall workload remained constant as the number of processors was varied, thus enabling one to examine the scalability of the speed-up and efficiency of the algorithms for a given problem size with increasing number of processors. The other parameters for the parallel stochastic hill-climbing algorithms were identical to those of their serial counterparts. In the NILM PSA and NILM PMCA algorithms, each process was independently terminated when the number of successful perturbations in any annealing step for that process equaled 0. In the PILM PSA and PILM PMCA algorithms, each process was terminated when the number of successful perturbations in an annealing step equalled 0 for *all* the processes *i.e.*, the

TABLE 3

Run-time results (in minutes) for the parallel conjugate gradient descent algorithm

n	Steepest descent: $N_{proc} = 1$	Parallel conjugate gradient descent			
		$N_{proc} = 1$	$N_{proc} = 2$	$N_{proc} = 4$	$N_{proc} = 8$
100	28.82	17	9.27	13.80	19.95
200	49.62	38.57	33.90	21.13	26.05
300	834.75	301.23	215.57	136.72	121.82

N_{proc}, number of processors; n, number of probes.

master process and all the child processes. This condition was checked during the synchronization phase at the end of each annealing step.

The parallel conjugate gradient descent algorithm was tested on artificial data sets with a varying number of probes ($n = 100$, 200, and 300). Table 3 shows the run-time results for the parallel conjugate gradient descent algorithm with varying number of processors N_{proc} and varying number of probes n. Table 3 also shows the run-time results for the serial steepest descent algorithm on the same data sets. As can be seen from Table 3, the serial version of the conjugate gradient descent algorithm is significantly faster than the serial version of the steepest descent algorithm. Moreover, Table 3 also shows how the performance of the parallel conjugate gradient descent algorithm scales with an increasing number of processors and increasing problem size.

The speed-up results for Table 3 are shown in Table 4. For $n = 100$, the best speed-up is attained for $N_{proc} = 2$ with a degradation in speed-up for $N_{proc} > 2$. For $n = 200$, the results show the best speed-up for $N_{proc} = 4$ with a degradation in speed-up for $N_{proc} > 4$. As for $n = 300$, the best speed-up is seen for $N_{proc} = 8$. These results are in conformity with expectations, since the interprocessor communication overhead and synchronization overhead tend to increasingly dominate the overall execution time with increasing number of processors N_{proc} for a problem of given size (*i.e.*, for a given value of n). The interprocessor communication overhead and synchronization overhead as a percentage of the computational load are higher for smaller problem sizes, which explains why the maximum speed-up obtained is for values of $N_{proc} < 8$. In summary, the payoff in the parallelization of the conjugate gradient descent algorithm is realized only for large values of n (*i.e.*, large problem sizes). This is a natural consequence of the network latency inherent in PVM systems that are comprised of a network of workstations.

The total number of iterations of the serial and parallel versions (with different N_{proc} values) of the conjugate gradient descent algorithm prior to convergence was also examined (Table 5). As can be seen in Table 5, the number of iterations prior to convergence varies significantly between the serial and the parallel versions of the conjugate gradient descent algorithm. On closer examination, we were able to conclude that this was due to numerical errors introduced by the parallelization process. Since the objective function is mathematically involved, and since the order in which numbers are added in the serial implementation and each of the parallel versions (with different N_{proc} values) is different, a numerical error is introduced. This error is the difference in the objective function values computed by the serial and parallel versions of the conjugate gradient descent algorithm in a given iteration having started from the same initial point. It was noted that the numerical error is initially insignificant. However, due to the iterative nature of the conjugate gradient descent algorithm, the results of the ith iteration depend entirely on the results of the $(i - 1)$th iteration, thereby causing the error to build up throughout the computation. Also, to ensure convergence to a global minimum very high-precision thresholds have been used to test for convergence. For instance, the bisection method terminates

TABLE 4

Speed-up results for the parallel conjugate gradient descent algorithm from Table 3

n	Parallel conjugate gradient descent			
	$N_{proc} = 1$	$N_{proc} = 2$	$N_{proc} = 4$	$N_{proc} = 8$
100	1.00	1.83	1.23	0.85
200	1.00	1.14	1.83	1.48
300	1.00	1.40	2.20	2.47

N_{proc}, number of processors; n, number of probes.

TABLE 5

Total number of iterations before convergence for the serial and parallel conjugate gradient descent algorithms

n	Conjugate gradient descent			
	$N_{proc} = 1$	$N_{proc} = 2$	$N_{proc} = 4$	$N_{proc} = 8$
100	231	138	200	173
200	260	306	271	285
300	594	552	576	578

N_{proc}, number of processors; n, number of probes.

TABLE 6

Average number of iterations per minute for the serial and parallel conjugate gradient descent algorithms

	Conjugate gradient descent			
n	$N_{proc} = 1$	$N_{proc} = 2$	$N_{proc} = 4$	$N_{proc} = 8$
100	13.59	14.89	14.49	8.67
200	6.74	9.03	12.83	10.94
300	1.97	2.56	4.21	4.74

N_{proc}, number of processors; n, number of probes.

TABLE 8

Speed-up results for the parallel conjugate gradient descent algorithm from Table 7

	Conjugate gradient descent			
n	$N_{proc} = 1$	$N_{proc} = 2$	$N_{proc} = 4$	$N_{proc} = 8$
100	1.00	1.10	1.07	0.64
200	1.00	1.34	1.90	1.62
300	1.00	1.30	2.14	2.41

N_{proc}, number of processors; n, number of probes.

once the value of the objective function varies by $<10^{-10}$. The conjugate gradient descent algorithm itself is deemed to have converged to a global minimum when the value of the objective function varies by $<10^{-5}$ (if the gradient vector has not vanished by then). Consequently, all three factors, *i.e.*, the different order of execution of arithmetic operations, the data dependency between successive iterations, and the high-precision thresholds, contribute to the discrepancy in the number of iterations prior to convergence between the serial and parallel versions (with different N_{proc} values) of the conjugate gradient descent algorithm. The final results of the serial and the parallel versions of the conjugate gradient descent algorithm are equivalent up to a precision of 10^{-4}.

Table 6 shows the average number of iterations per minute for both the serial and parallel versions of the conjugate gradient descent algorithm. This information is derived from the overall execution time (Table 3) and the number of iterations prior to convergence (Table 5). Table 7 shows the average time (in seconds) per iteration for the serial and parallel versions (with different N_{proc} values) of the conjugate gradient descent algorithm. Table 8 shows the speed-up values for the parallel conjugate gradient descent algorithm for varying values of N_{proc} based on the results in Table 7. Although the speed-up results in Table 8 compare well with those in Table 4, one can observe greater uniformity and consistency in the results presented in Table 8. The speed-up curves corresponding to the speed-up results in Tables 4 and 8 are shown in Figures 11 and 12, respectively.

In the case of the parallel stochastic hill-climbing algorithms, experiments were conducted on problem sizes of $n = 10$ and $n = 30$ probes. All four algorithms—NILM PSA, PILM PSA, NILM PMCA, and PILM PMCA—were tested using the MLE objective function. Since the value of n (*i.e.*, problem size) is small, the serial version of the conjugate gradient descent algorithm was used. The run-time results for these algorithms are shown in Table 9. It can be observed that the serial and parallel versions of the MCA algorithm took longer to converge when compared to their SA counterparts. As Table 10 shows, the average run time per annealing step of the serial or any of the parallel versions (with different N_{proc} values) of the SA algorithm is higher than that of its MCA counterpart. The higher run time of the PMCA algorithms is attributable to the fact that they took a larger number of iterations to converge to a globally optimal solution.

The speed-up results for the PSA and PMCA algorithms are shown in Table 11. The corresponding speed-up curves for $n = 10$ and $n = 30$ are shown in Figures

TABLE 7

Average time (in seconds) per iteration for the serial and parallel conjugate gradient descent algorithms

	Conjugate gradient descent			
n	$N_{proc} = 1$	$N_{proc} = 2$	$N_{proc} = 4$	$N_{proc} = 8$
100	4.42	4.03	4.14	6.92
200	8.90	6.65	4.68	5.48
300	30.43	23.43	14.24	12.65

N_{proc}, number of processors; n, number of probes.

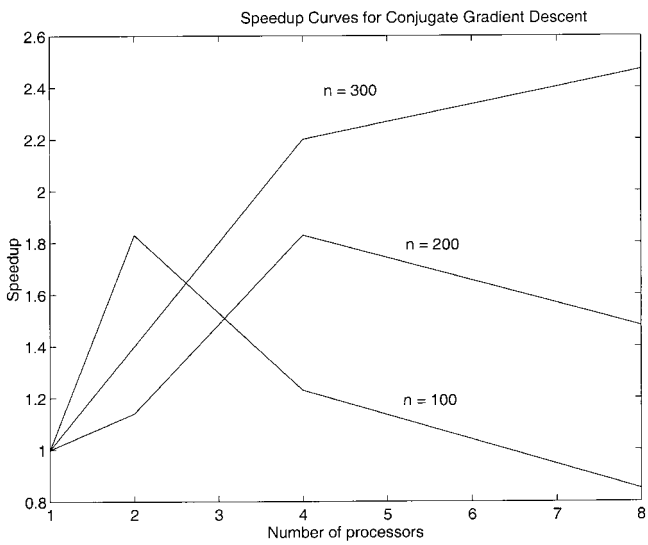

FIGURE 11.—Speed-up curves for the parallel conjugate gradient descent algorithm from Table 4.

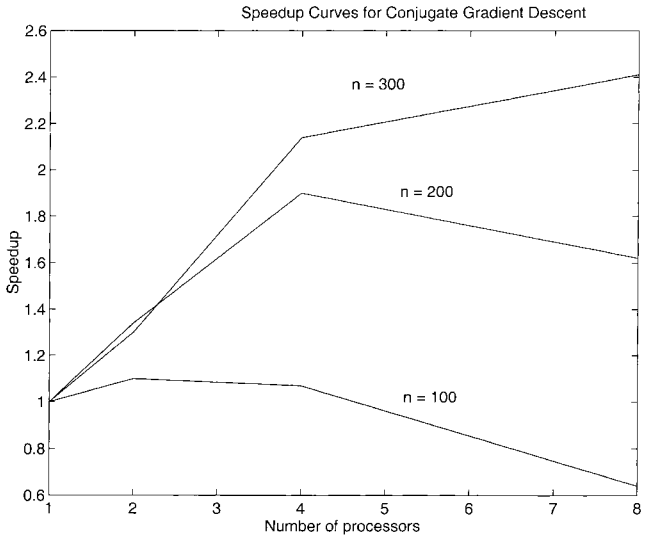

FIGURE 12.—Speed-up curves for the parallel conjugate gradient descent algorithm from Table 8.

13 and 14, respectively. As can be observed, the PSA and PMCA algorithms exhibit consistent and scalable speed-up with increasing N_{proc} values. As expected, the speed-up scales better with increasing N_{proc} values for larger problem sizes (*i.e.*, larger values of n). The PSA and PMCA algorithms arrived at the correct probe ordering in all cases but for one exception. In the case $n = 30$ and $N_{\text{proc}} = 8$, the PMCA algorithm came up with the reverse probe ordering (*i.e.*, Π^R instead of Π), which is expected since the likelihood function is unique only up to reversal in the probe ordering. Consequently, the MLE procedure is capable of recovering the correct probe ordering only up to reversal.

The absolute root mean squared error (RMSE) χ between the true interprobe spacings Y and the estimated interprobe spacings $\hat{Y}$ is defined as

$$\chi = \sqrt{\frac{|Y - \hat{Y}|^2}{n}}. \tag{47}$$

The RMSE value can also be expressed as a percentage of N, the total length of the chromosome. The absolute and percentage RMSE values for the PSA and PMCA algorithms for $n = 10$ and $n = 30$ are shown in Table 12. From the statistical theory underlying the maximum-likelihood estimation procedure it can be shown that the percentage RMSE value asymptotically approaches 0 in the limit $n \to \infty$ (LEHMAN 1983; HOGG and CRAIG 1995). This trend can be observed in the percentage RMSE values in Table 12.

Experimental results on real data: The parallel implementation of the MLE-based physical mapping algorithm was also tested on real probe-clone hybridization data from linkage group VII of the fungal genome *N. crassa*. The data were comprised of 105 probes and 1678 clones. The physical map was generated by first ordering the probes and estimating the interprobe distances using the MLE procedure. The intervening clones were then inserted between pairs of successive probes to generate a physical map of the chromosome. The resulting physical map exhibiting a total of 39 contig breaks can be viewed on the GENETICS web site at http://www.genetics.org/supplemental/. The 1's in the physical map that do not conform to the consecutive 1's property (BOOTH and LUEKER 1976) are identified as false positives and replaced by *'s. However, due to the inherent sparsity of the clone-probe hybridization matrix H, it is not possible to identify the false negatives (CHRISTOF and KECECIOGLU 1999). A parallel version of the MLE-based physical mapping procedure that used the NILM PSA algorithm in conjunction with the serial CGD algorithm exhibited a speed-up of 1.5 on a cluster of three SUN UltraSparc1 workstations (350 MHz, 128 MB RAM) connected via 100 Mbs fast Ethernet. Although the results are encouraging, the speed-up of the parallel MLE-based physical mapping procedure leaves room for improvement.

CONCLUSIONS AND DISCUSSION

In this article, a MLE-based approach to physical map reconstruction under a probabilistic model of hybridiza-

TABLE 9

Run-time results (in minutes) for the PSA and PMCA algorithms

Algorithm	n	$N_{\text{proc}} = 1$	$N_{\text{proc}} = 2$	$N_{\text{proc}} = 4$	$N_{\text{proc}} = 8$
NILM PSA	10	33.48	17.28	9.90	5.60
PILM PSA		33.48	17.42	9.93	5.76
NILM PMCA		38.43	19.32	10.35	5.79
PILM PMCA		38.43	19.47	10.43	5.88
NILM PSA	30	198.57	100.56	52.92	28.72
PILM PSA		198.57	102.43	53.77	29.11
NILM PMCA		217.72	110.65	56.24	30.85
PILM PMCA		217.72	113.58	58.30	32.06

N_{proc}, number of processors; n, number of probes.

TABLE 10

Average time per annealing step T_{step} (in minutes) for the PSA and PMCA algorithms

Algorithm	n	T_{step}
NILM PSA		16.40
PILM PSA	10	16.40
NILM PMCA		12.83
PILM PMCA		12.83
NILM PSA		49.17
PILM PSA	30	49.17
NILM PMCA		38.77
PILM PMCA		38.77

n, number of probes.

tion errors consisting of false positives and false negatives was presented. The maximum-likelihood estimator optimizes a likelihood function defined by the ordering of probes and interprobe spacings under an experimental protocol wherein clones of equal length are hybridized to a maximal subset of nonoverlapping equal-length clones termed probes. The estimation procedure was shown to involve a combination of continuous and discrete optimization, the former to determine a set of optimal interprobe spacings for a given probe ordering and the latter to determine the optimal probe ordering. The conjugate gradient descent procedure was used to determine the optimal spacings between probes for a given probe ordering. The optimal probe ordering was determined using stochastic combinatorial optimization procedures such as SA and MCA.

The problem of MLE-based physical map reconstruction in the presence of errors is a problem of high computational complexity, thus providing the motivation for parallel computing. A two-level parallelization strategy was proposed wherein the conjugate gradient descent procedure was parallelized at the lower level, and the stochastic hill-climbing algorithm was simultaneously parallelized at the higher level. The parallel algorithms were implemented on a distributed-memory multiprocessor cluster running the PVM environment. A data parallel approach, where the components of the gradient vector are distributed among the individual processors in the PVM system, was deemed more suitable for the parallelization of the conjugate gradient descent procedure. A control parallel scheme where individual processors perform noninteracting or periodically interacting searches was deemed more suitable for the parallelization of the combinatorial stochastic hill-climbing procedures.

Experimental results on artificial clone-probe data showed that the payoff in data parallelization of the conjugate gradient descent procedure was realized only for large problem sizes (*i.e.*, large values of n). A similar trend was observed in the case of the parallel stochastic hill-climbing algorithms. In all cases, the parallel implementation of the maximum-likelihood estimator resulted in the correct probe ordering, except in one case wherein the probe ordering was reversed. The RMSE between the resulting interprobe distances and the true interprobe distances was computed. The percentage RMSE was seen to exhibit a decreasing trend with increasing problem values of n (*i.e.*, problem size), which is in conformity with the statistical theory underlying the MLE procedure. Experimental results on real hybridization data from linkage group VII of the fungal genome *N. crassa* yielded a physical map with 39 contig breaks. The parallel MLE-based physical mapping procedure that used a combination of NILM PSA and the serial CGD algorithm exhibited a speed-up of 1.5 on a three-workstation cluster interconnected via 100 Mbs fast Ethernet.

The formulation of the ML model entailed certain key assumptions. The assumption of a constant probe/clone size is reasonable (Kelkar *et al.* 2001). The assumption of a uniform distribution of clones along the length of the chromosome is also reasonable since

TABLE 11

Speed-up results for the PSA and PMCA algorithms

Algorithm	n	$N_{proc} = 1$	$N_{proc} = 2$	$N_{proc} = 4$	$N_{proc} = 8$
NILM PSA		1.0	1.94	3.38	5.98
PILM PSA	10	1.0	1.92	3.37	5.81
NILM PMCA		1.0	1.99	3.71	6.64
PILM PMCA		1.0	1.97	3.68	6.65
NILM PSA		1.0	1.97	3.75	6.91
PILM PSA	30	1.0	1.94	3.69	6.82
NILM PMCA		1.0	1.97	3.87	7.06
PILM PMCA		1.0	1.92	3.73	6.79

N_{proc}, number of processors; n, number of probes.

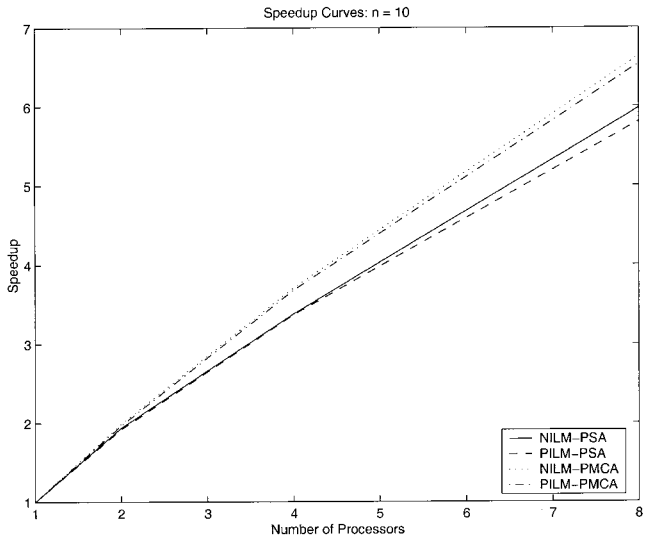

FIGURE 13.—Speed-up curves for the parallel stochastic hill-climbing algorithms for $n = 10$ from Table 9.

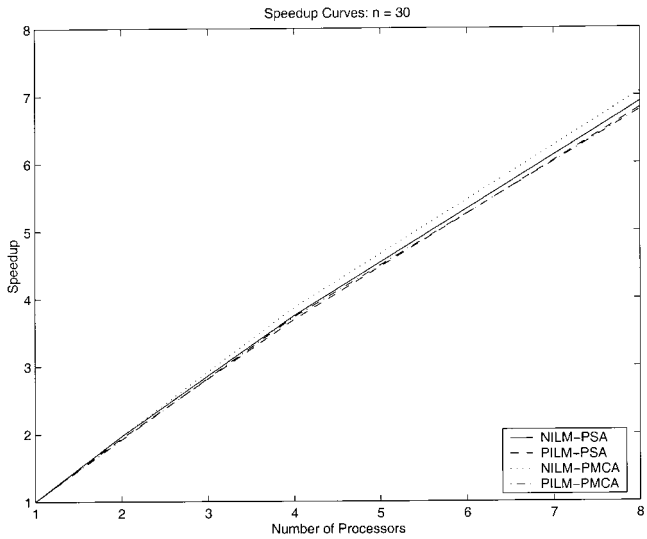

FIGURE 14.—Speed-up curves for the parallel stochastic hill-climbing algorithms for $n = 30$ from Table 9.

KELKAR *et al.* (2001) and AIGN *et al.* (2001) have taken substantial effort to represent fragments of *N. crassa* DNA in multiple cloning vectors. The most difficult assumption is that of a uniform distribution of probes along the length of the chromosome due to the presence of repeats (XIONG *et al.* 1996). This has not been a serious problem in the case of *N. crassa* thus far, due to the paucity of off-diagonal hybridization signals in the physical maps (AIGN *et al.* 2001; HALL *et al.* 2001). However, in general, the presence of DNA repeats does pose a serious problem in physical map creation and the ML model will need to be extended to address this issue. The ML model will also need to be enhanced to take into account hybridization errors due to deletions and chimerism, especially when longer length clones (and probes) are used in the sampling-without-replacement protocol. Another key assumption in the computation of the ML estimate is that the n-dimensional space of interprobe spacings is convex or can be decomposed into a finite number of convex regions for a given probe ordering. This assumption permits the use of local search-based techniques to determine a local minimum (which also happens to be a global minimum) of the ML objective function for a given probe ordering. This assumption is valid only when the clone coverage is sufficiently large. In the absence of this assumption, the problem of determining a global minimum of the ML objective function for a given probe ordering becomes intractable.

Future research will attempt to improve the performance of the parallel MLE-based physical mapping procedure on real hybridization data. Future research will also investigate extensions and enhancements to the ML objective function to account for errors due to repeat DNA sequences, deletions, and chimerism. The consistency of the ML estimator needs to be rigorously analyzed. This would entail a proof of the asymptotic convergence of the inferred physical map to the true physical map with probability one as the number of probes (and clones) grows. Statistical methods need to

TABLE 12

Root mean squared error (RMSE) values for the interprobe spacings

Algorithm	n	$N_{proc} = 1$ (%)	$N_{proc} = 2$ (%)	$N_{proc} = 4$ (%)	$N_{proc} = 8$ (%)
NILM PSA	10	2.712 (1.51)	2.712 (1.51)	2.712 (1.51)	2.712 (1.51)
PILM PSA		2.712 (1.51)	2.712 (1.51)	2.712 (1.51)	2.712 (1.51)
NILM PMCA		4.736 (2.63)	4.736 (2.63)	2.712 (1.51)	4.695 (2.61)
PILM PMCA		4.736 (2.63)	4.736 (2.63)	2.712 (1.51)	4.695 (2.61)
NILM PSA	30	2.306 (0.34)	2.306 (0.34)	2.306 (0.34)	5.359 (0.79)
PILM PSA		2.306 (0.34)	2.306 (0.34)	2.306 (0.34)	5.359 (0.79)
NILM PMCA		4.029 (0.59)	4.029 (0.59)	2.306 (0.34)	2.306 (0.34)
PILM PMCA		4.029 (0.59)	4.029 (0.59)	2.306 (0.34)	2.306 (0.34)

N_{proc}, number of processors; n, number of probes.

be developed to assess the statistical reliability of the ML estimator. The rate of convergence of the inferred physical map to the true physical map could be used to establish asymptotic confidence levels for links between pairs of probes (or clones) on the physical map (Xiong *et al.* 1996). Variational analysis will provide the tools for an asymptotic analysis and bootstrap resampling techniques (Wang *et al.* 1994b) could provide a tool for estimating the statistical measure of confidence in the links on the physical map. A rigorous sensitivity analysis is necessary to show the validity and robustness of the various assumptions underlying the ML model. Extensions to the ML objective function to be able to integrate ordinal information from probes containing markers anchored to genetic maps (Hall *et al.* 2001) also need to be investigated. The current PVM implementation of the ML estimator is targeted toward a homogeneous distributed processing platform such as a network of identical workstations. Future research will explore and address issues that deal with the parallelization of the ML estimator on a heterogeneous distributed processing platform such as a network of workstations that differ in processing speeds.

The authors thank Dr. David Lowenthal for access to his PVM workstation cluster. This research was supported in part by a National Research Initiative Competitive Grants Program (NRICGP) grant by the U.S. Department of Agriculture to Dr. Suchendra M. Bhandarkar.

LITERATURE CITED

Aarts, E. H. L., and K. Korst, 1989 *Simulated Annealing and Boltzman Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. Wiley, New York.

Aarts, E. H. L., F. M. J. De Bont, J. H. A. Habers and P. J. M. Van Laarhoven, 1986 A parallel statistical cooling algorithm, pp. 87–97 in *Lecture Notes in Computer Science: Proceedings of the 3rd Annual Symposium on Theoretical Aspects of Computer Science*, Vol. 210. Springer-Verlag, Berlin.

Aign, V., U. Schulte and J. D. Hoheisel, 2001 Hybridization mapping of *Neurospora crassa* linkage groups II and V. Genetics **157:** 1015–1020.

Alizadeh, F., R. M. Karp, D. K. Weisser and G. Zweig, 1994 Physical mapping of chromosomes using unique probes, pp. 489–500 in *Proceedings of the ACM-SIAM Conference on Discrete Algorithms*. ACM Press, New York.

Alizadeh, F., R. M. Karp, L. A. Newberg and D. K. Weisser, 1995 Physical mapping of chromosomes: a combinatorial problem in molecular biology. Algorithmica **13**(1/2): 52–76.

Arnold, J., 1997 Editorial. Fungal Genet. Biol. **21**(3): 254–257.

Arnold, J., and M. T. Cushion, 1997 Constructing a physical map of the *Pneumocystis* genome. J. Eukaryot. Microbiol. **44:** 8S.

Arratia, R., E. S. Lander, S. Tavare and M. S. Waterman, 1991 Genomic mapping by anchoring random probes: a mathematical analysis. Genomics **11:** 806–827.

Azencott, R. (Editor), 1992 *Simulated Annealing: Parallelization Techniques*. Wiley, New York.

Balding, D. J., 1994 Design and analysis of chromosome physical mapping experiments. Philos. Trans. R. Soc. Lond. Ser. B **334:** 329–335.

Banerjee, P., M. H. Jones and J. S. Sargent, 1990 Parallel simulated annealing algorithms for cell placement on the hypercube multiprocessor. IEEE Trans. Parallel Distributed Syst. **1:** 91–106.

Ben-Dor, A., and B. Chor, 1997 On constructing radiation hybrid maps, pp. 17–26 in *Proceedings of the ACM Conference on Computational Molecular Biology*. ACM Press, New York.

Bennett, J. W., 1997 White paper: genomics for filamentous fungi. Fungal Genet. Biol. **21**(1): 3–7.

Bhandarkar, S. M., 1997 Parallel processing for chromosome reconstruction from physical maps—a case study of MIMD parallelism on the hypercube. Parallel Algorithms and Applications **12:** 231–252.

Bhandarkar, S. M., and S. Machaka, 1997 Chromosome reconstruction from physical maps using a cluster of workstations. J. Supercomput. **11**(1): 61–86.

Bhandarkar, S. M., S. Chirravuri and J. Arnold, 1996a Parallel computing of physical maps—a comparative study in SIMD and MIMD parallelism. J. Comput. Biol. **3**(4): 503–528.

Bhandarkar, S. M., S. Chirravuri and J. Arnold, 1996b PARODS—A study of parallel algorithms for ordering DNA sequences. Int. J. Comput. Appl. Biosci. **12**(4): 269–280.

Bhandarkar, S. M., S. Chirravuri, S. Machaka and J. Arnold, 1998 Parallel computing for chromosome reconstruction via ordering of DNA sequences. Parallel Comput. **24**(8): 1177–1204.

Bhanot, G., M. Creutz and H. Neuberger, 1984 Microcanonical simulation of Ising systems. Nuclear Phy. **B235** (FS11): 417–434.

Booth, K. S., and G. S. Lueker, 1976 Testing for the consecutive one's property, interval graphs and graph planarity using pq-tree algorithms. J. Comput. Systems Sci. **13:** 335–379.

Brody, H., J. Griffith, A. J. Cuticchia, J. Arnold and W. Timberlake, 1991 Chromosome-specific recombinant libraries from the fungus *Aspergillus nidulans*. Nucleic Acids Res. **19:** 3105–3109.

Casotto, A., F. Romeo and A. Sangiovanni-Vincentelli, 1987 A parallel simulated annealing algorithm for the placement of macro cells. IEEE Trans. Computer-Aided Design **1:** 838–847.

Chor, B., and M. Sudan, 1995 A geometric approach to betweenness, pp. 227–237 in *Proceedings of the European Symposium on Algorithms: Springer-Verlag Lecture Notes in Computer Science*, Vol. 979. Springer-Verlag, Berlin.

Christof, T., and J. D. Kececioglu, 1999 Computing physical maps of chromosomes with non-overlapping probes by branch-and-cut. Proceedings of the ACM Conference on Computational Molecular Biology, Lyon, France, pp. 115–123.

Christof, T., M. Jünger, J. D. Kececioglu, P. Mutzel and G. Reinelt, 1997 A branch-and-cut to physical mapping of chromosomes by unique end probes. J. Comput. Biol. **4**(4): 433–447.

Creutz, M., 1983 Microcanonical Monte Carlo simulation. Phys. Rev. Lett. **50**(19): 1411–1414.

Cuticchia, A. J., J. Arnold and W. E. Timberlake, 1992 The use of simulated annealing in chromosome reconstruction experiments based on binary scoring. Genetics **132:** 591–601.

Cuticchia, A. J., J. Arnold and W. E. Timberlake, 1993 ODS: ordering DNA sequences—a physical mapping algorithm based on simulated annealing. Comput. Appl. Biosci. **9**(2): 215–219.

Dorny, C. N., 1980 *A Vector Space Approach to Models and Optimization*. R. E. Krieger, Huntington, NY.

Fasulo, D. P., T. Jiang, R. M. Karp, R. Settergren and E. C. Thayer, 1997 An algorithmic approach to multiple complete digest mapping, pp. 118–127 in *Proceedings of the ACM Conference on Computational Molecular Biology*. ACM Press, New York.

Fu, Y. X., W. E. Timberlake and J. Arnold, 1992 On the design of genome mapping experiments using short synthetic oligo nucleotides. Biometrics **48:** 337–359.

Garey, M. S., and D. S. Johnson, 1979 *Computers and Intractability: A Guide to the Theory of NP—Completeness*. W. H. Freeman, New York.

Geist, A., A. Beguelin, J. Dongarra, W. Jiang, R. Mancheck *et al.*, 1994 *PVM Parallel Virtual Machine—A User's Guide and Tutorial for Networked Parallel Computing*. MIT Press, Cambridge, MA.

Geman, S., and D. Geman, 1984 Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images. IEEE Trans. Pattern Anal. Mach. Intell. **6:** 721–741.

Greening, D. R., 1990 Parallel simulated annealing techniques. Physica D **42:** 293–306.

Greenberg, D. S., and S. Istrail, 1995 Physical mapping by STS hybridization: algorithmic strategies and the challenge of software evaluation. J. Comput. Biol. **2**(2): 219–273.

Hadamard, H., 1923 *Lectures on the Cauchy Problem in Linear Partial Differential Equations*. Yale University Press, New Haven, CT.

Hall, R. D., S. M. Bhandarkar and J. Wang, 2001 ODS2: a multiplatform software application for creating integrated physical and genetic maps. Genetics **157:** 1045–1056.

Hestenes, M. R., 1980 *Conjugate Direction Methods in Optimization*. Springer-Verlag, New York.

Hestenes, M. R., and E. Stiefel, 1952 Methods of conjugate gradient for solving linear systems. J. Res. Natl. Bureau Standards **49:** 409–436.

Hogg, R. V., and A. T. Craig, 1995 *Introduction to Mathematical Statistics*, Ed. 5. Prentice-Hall, Englewood Cliffs, NJ.

Jain, M., and E. W. Myers, 1997 Algorithms for computing and integrating physical maps using unique probes. J. Comput. Biol. **4**(4): 449–466.

Jayaraman, R., and R. Rutenbar, 1987 Floor planning by annealing ona hypercube multiprocessor, pp. 346–349 in *Proceedings of the IEEE International Conference on Computer-Aided Design*. IEEE Press, Washington, DC.

Jiang, T., and R. M. Karp, 1997 Mapping clones with a given ordering or interleaving, pp. 400–409 in *Proceedings of the ACM-SIAM Conference on Discrete Algorithms*. ACM Press, New York.

Karp, R. M., and R. Shamir, 1998 Algorithms for optical mapping, pp. 117–124 in *Proceedings of the ACM Conference on Computational Molecular Biology*. ACM Press, New York.

Kececioglu, J. D., and E. W. Myers, 1995 Combinatorial algorithms for DNA sequence assembly. Algorithmica **13:** 7–51.

Kececioglu, J. D., S. S. Shete and J. Arnold, 2000 Reconstructing distances in physical maps of chromosomes with nonoverlapping probes. Proceedings of the ACM Conference on Computational Molecular Biology, Tokyo, Japan, pp. 183–192.

Kelkar, H. S., J. Griffith, M. E. Case, S. F. Covert, R. D. Hall *et al.*, 2001 The *Neurospora crassa* genome: cosmid libraries sorted by chromosome. Genetics **157:** 979–990.

Kim, Y., and M. Kim, 1990 A stepwise overlapped parallel annealing algorithm on a message passing multiprocessor system. Concurrency: Practice Experience **2**(2): 123–148.

Kincaid, D., and W. Cheney, 1991 *Numerical Analysis Mathematics of Scientific Computing*. Brooks/Cole, Pacific Grove, CA.

Kirkpatrick, S., C. Gelatt Jr. and M. Vecchi, 1983 Optimization by simulated annealing. Science **220:** 498–516.

Lander, E. S., and P. Green, 1987 Construction of multi-locus genetic linkage maps in humans. Proc. Natl. Acad. Sci. USA **84:** 2363–2367.

Lander, E. S., and M. S. Waterman, 1988 Genomic mapping by fingerprinting random clones: a mathematical analysis. Genomics **2:** 231–239.

Lee, F. H., 1995 Parallel simulated annealing on a message-passing multicomputer. Ph.D. dissertation, Department of Electrical Engineering, Utah State University, Logan, UT.

Lee, J. K., V. Dancik and M. S. Waterman, 1998 Estimation for restriction sites observed by optical mapping using reversible-jump Markov chain Monte Carlo, pp. 147–152 in *Proceedings of the ACM Conference on Computational Molecular Biology*. ACM Press, New York.

Lehman, E. L., 1983 *Theory of Point Estimation*. Wiley, New York.

Metropolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller, 1953 Equation of state calculations by fast computing machines. J. Chem. Phys. **21:** 1087–1092.

Mizukami, T., W. I. Chang, I. Garkatseve, N. Kaplan, D. Lombardi *et al.*, 1993 A 13Kb resolution cosmid map of the 14 Mb fission yeast genome by nonrandom sequence-tagged site mapping. Cell **73:** 121–132.

Mott, R., A. V. Grigoriev, E. Maier, J. D. Hoheisel and H. Lehrach, 1993 Algorithms and software tools for ordering clone libraries: application to the mapping of the genome *S. pombe*. Nucleic Acids Res. **21**(8): 1965–1974.

Muthukrishnan, S., and L. Parida, 1997 Towards constructing physical maps by optical mapping: an effective, simple, combinatorial approach, pp. 209–219 in *Proceedings of the ACM Conference on Computational Molecular Biology*. ACM Press, New York.

Nelson, D. O., and T. P. Speed, 1994 Statistical issues in constructing high resolution physical maps. Stat. Sci. **9:** 334–354.

Polak, E., 1997 Optimization: algorithms and consistent approximations. Appl. Math. Sci. **124:** 320–345.

Prade, R. A., J. Griffith, K. Kochut, J. Arnold and W. E. Timberlake, 1997 *In vitro* reconstruction of the *Aspergillus nidulans* genome. Proc. Natl. Acad. Sci. USA **94:** 14564–14569.

Press, W. H., B. P. Flannery, S. A. Teukolsky and W. T. Vetterling, 1988 *Numerical Recipes in C*. Cambridge University Press, New York.

Romeo, F., and A. Sangiovanni-Vincentelli, 1991 A theoretical framework for simulated annealing. Algorithmica **6:** 302–345.

Shete, S. S., 1998 Estimation problems in physical mapping of a chromosome and in a branching process with immigration. Ph.D. dissertation, Department of Statistics, The University of Georgia, Athens, GA.

Slonim, D., L. Kruglyak, L. Stein and E. Lander, 1997 Building human genome maps with radiation hybrids, pp. 277–286 in *Proceedings of the ACM Conference on Computational Molecular Biology*. ACM Press, New York.

Sturtevant, A. H., 1913 The linear arrangement of six sex-linked factors in *Drosophila* as shown by their mode of association. J. Exp. Zool. **14:** 43–49.

Sunderam, V., 1990 PVM: a framework for parallel distributed computing. Concurrency: Practice Experience **2**(2): 315–339.

Wang, Y., R. A. Prade, J. Griffith, W. E. Timberlake and J. Arnold, 1994a A fast random cost algorithm for physical mapping. Proc. Natl. Acad. Sci. USA **91:** 11094–11098.

Wang, Y., R. A. Prade, J. Griffith, W. E. Timberlake and J. Arnold, 1994b ODS-BOOTSTRAP: assessing the statistical reliability of physical maps by bootstrap resampling. Comput. Appl. Biosci. **10:** 625–635.

Wilson, D. B., D. S. Greenberg and C. A. Phillips, 1997 Beyond islands: runs in clone-probe matrices, pp. 320–329 in *Proceedings of the ACM Conference on Computational Molecular Biology*. ACM Press, New York.

Witte, E. E., R. D. Chamberlain and M. A. Franklin, 1991 Parallel simulated annealing using speculative computation. IEEE Trans. Parallel Distributed Syst. **2**(4): 483–494.

Wong, C. P., and R. D. Fiebrich, 1987 Simulated annealing-based circuit placement on the connection machine system, pp. 78–82 in *Proceedings of the International Conference on Computer Design*. IEEE Press, Washington, DC.

Xiong, M., H. J. Chen, R. A. Prade, Y. Wang, J. Griffith *et al.*, 1996 On the consistency of a physical mapping method to reconstruct a chromosome *in vitro*. Genetics **142**(1): 267–284.

Zhang, M. Q., and T. G. Marr, 1993 Genome mapping by nonrandom anchoring: a discrete theoretical analysis. Proc. Natl. Acad. Sci. USA **90:** 600–604.

Communicating editor: J. Arnold