

# A Parallel Evolutionary Method for Physical Mapping of Chromosomes

Suchendra M. Bhandarkar<sup>1\*</sup> and Jinling Huang<sup>1</sup>

Jonathan Arnold<sup>2</sup>

<sup>1</sup>Department of Computer Science

<sup>2</sup>Department of Genetics

The University of Georgia

Athens, Georgia 30602-7404, USA

E-mail: suchi@cs.uga.edu

## 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. A parallel evolutionary method for a maximum likelihood estimation-based approach to physical map reconstruction is presented. The estimation procedure entails gradient descent search for determining the optimal spacings between probes for a given probe ordering. The optimal probe ordering is determined using an evolutionary algorithm. A two-tier parallelization strategy is proposed wherein the gradient descent search is parallelized at the lower level and the evolutionary algorithm is simultaneously parallelized at the higher level. Implementation and experimental results on a network of shared-memory symmetric multiprocessors (SMPs) are presented. The evolutionary algorithm is seen to result in physical maps with fewer contig breaks when compared to simulated Monte Carlo algorithms such as simulated annealing and the large-step Markov chain algorithm.*

**Keywords:** Physical Mapping, Evolutionary Algorithms, Genetic Algorithms, Maximum Likelihood Estimation, Chromosome Reconstruction, Gradient Descent Search

## 1 Introduction

Generation of entire chromosomal maps is a central problem in genetics. Chromosomal maps fall into two broad categories - *genetic maps* and *physical maps*. Genetic maps are typically of low resolution (1–10 million base pairs (Mb)) and represent an ordering of genetic markers along a chromosome where the distance between two genetic markers is related to their recombination frequency. A physical map is an ordering of distinguishable DNA fragments called *clones* or *contigs* by their position along the entire chromosome where the clones may or may not contain genetic markers. A physical map has a much higher resolution (10–100 thousand base pairs (Kb)) than a genetic map of the same chromosome. 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 direct recovery and molecular manipulation of genes of interest.

The physical mapping protocol essentially determines the nature of clonal data and the probe selection procedure. The physical mapping protocol used in this project is the one based on

---

\*Author for correspondence. E-mail: suchi@cs.uga.edu, Tel: (706) 542-1082, Fax: (706) 542-2966

*sampling without replacement* [3]. Under this protocol, a maximal set  $\mathcal{P}$  of non-overlapping equal-length clones from a library is selected as the probe set. The remaining clones  $\mathcal{C}$  in the library are hybridized to the probe set resulting in a digital hybridization signature for each clone. The clone-probe overlap pattern is represented by a binary hybridization matrix  $H$  where  $H_{ij} = 1$  if the  $i$ th clone hybridizes to the  $j$ th probe and  $H_{ij} = 0$  otherwise. If the probes in  $\mathcal{P}$  are ordered with respect to their position along a chromosome, then by selecting from  $H$  a common overlapping clone for each pair of adjacent probes, a minimal set of clones and probes that covers the entire chromosome (i.e., a minimal tiling) can be obtained [3]. 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 [9] can then be used to reconstruct the DNA sequence of the entire chromosome. In reality,  $H$  could be expected to contain false positives and false negatives.  $H_{ij}$  would be a false positive if  $H_{ij} = 1$  when in fact  $H_{ij} = 0$ . Conversely,  $H_{ij}$  would be a false negative if  $H_{ij} = 0$  when in fact  $H_{ij} = 1$ . In this paper, we confine ourselves to errors in the form of false positives and false negatives.

In this paper we briefly describe a maximum likelihood (ML) estimator proposed in [3, 10] which determines the ordering of probes in the probe set  $\mathcal{P}$  and also the inter-probe 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 inter-probe spacings for a particular probe ordering entails continuous optimization. We propose a two-tier parallelization strategy for efficient implementation of the above estimator. The upper-level comprises of parallel discrete optimization using an evolutionary method whereas the lower-level comprises of parallel conjugate gradient descent. The resulting parallel estimator is implemented on a network of shared-memory symmetric multiprocessors (SMPs) using a combination of the Message Passing Interface (MPI) environment [15] and multi-threaded programming [1]. Convergence, speedup and scalability characteristics of the parallel estimator are analyzed and discussed.

## 2 Mathematical Formulation of the ML Estimator

The ML estimator reconstructs the ordering of probes in the probe set  $\mathcal{P}$  and the inter-probe 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  $\mathcal{P} = \{P_1, P_2, \dots, P_n\}$  of  $n$  probes and a set  $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$  of  $k$  clones generated using the sampling-without-replacement protocol and the  $k \times n$  clone-probe hybridization matrix  $H$  containing both false positives and false negatives with predefined probabilities, reconstruct the correct ordering  $\Pi = (\pi_1, \pi_2, \dots, \pi_n)$  of the probes and also the correct spacing  $Y = (Y_1, Y_2, \dots, Y_n)$  between the probes such that the statistical likelihood of the observed hybridization matrix  $H$  is maximized [3, 10]. The ordering  $\Pi$  is a permutation of  $(1, \dots, n)$  that gives the labels (indices) of the probes in left-to-right order across the chromosome. In the inter-probe spacing vector  $Y$ ,  $Y_1$  denotes the space between the left end of the first probe  $P_{\pi_1}$  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  $P_{\pi_i}$  (where  $2 \leq i \leq n$ ). The spacing between the right end of probe  $P_{\pi_n}$  and the right end of the chromosome is given by  $Y_{n+1} = N - nM - \sum_{i=1}^n Y_i$  where  $N$  is length of the chromosome and  $M$  is the length of each clone/probe. Note that the protocol requires that all probes and clones be of the same length.

## 2.1 Mathematical Notation

The mathematical notation used in the formulation of the ML estimator is given below:  
 $N$  : Length of the chromosome,  $M$  : Length of a clone/probe,  $n$  : Number of probes,  $k$  : Number of clones,  $\rho$  : Probability of false positive,  $\eta$  : Probability of false negative,  $H = ((h_{i,j}))_{1 \leq i \leq k, 1 \leq j \leq n}$ : clone-probe hybridization matrix, where  $h_{i,j} = \begin{cases} 1 & \text{if clone } C_i \text{ hybridizes with probe } P_j \\ 0 & \text{otherwise,} \end{cases}$ ,  $H_i$  :  $i$ th row of the hybridization matrix,  $\Pi = (\pi_1, \dots, \pi_n)$ : permutation of  $\{1, 2, \dots, n\}$  which denotes the probe labels in the ordering when scanned from left to right along the chromosome,  $p_i = \sum_{j=1}^n h_{i,j}$ : number of 1's in  $H_i$ ,  $P = \sum_{i=1}^k p_i$ : total number of 1's in  $H$ ,  $Y = (Y_1, Y_2, \dots, Y_n)$ : vector of inter-clone spacings where  $Y_i$  is the spacing between the right end of  $P_{\pi_{i-1}}$  and the left end of  $P_{\pi_i}$  ( $2 \leq i \leq n$ ), and  $Y_1$  is the spacing between the left end of  $P_{\pi_1}$  and the left end of the chromosome, and  $\mathcal{F} \subseteq \mathcal{R}^n$ : set of feasible interprobe spacings  $Y = \{Y_1, \dots, Y_n\}$  such that  $Y_i \geq 0$ ,  $1 \leq i \leq n$  and  $N - nM - \sum_{i=1}^n Y_i \geq 0$ .

## 2.2 The ML Model

Given a vector of inter-probe spacings  $Y = (Y_1, \dots, Y_n)$ , there are  $2^{n+1}$  possible cases to consider depending on whether  $0 \leq Y_i \leq M$  or  $Y_i > M$  where  $0 \leq i \leq n + 1$ . It can be shown that the  $2^{n+1}$  cases can be analyzed based on the clone-probe overlap pattern [3]. In general, the clone-probe overlap pattern results in three different types of regions namely,

**Type 1:** The *Both* region  $R_B(P_{\pi_j}, P_{\pi_{j+1}})$  between probes  $P_{\pi_j}$  and  $P_{\pi_{j+1}}$ , for  $j = 1, \dots, n - 1$ . A clone hybridizes to both probes if its left end falls in this region.

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

**Type 3:** The *None* region  $R_N(P_{\pi_j})$  after probe  $P_{\pi_j}$ , for  $j = 0, \dots, n$ . A clone will hybridize to no probe if its left end falls in this region. Here probe  $P_{\pi_0}$  denotes the beginning of the chromosome. Let  $l(R)$  denote the length of region  $R$ . It can be shown that for  $j = 1, \dots, n - 1$ ,  $l(R_B(P_{\pi_j}, P_{\pi_{j+1}})) = M - \min(Y_{j+1}, M)$ , and for  $j = 1, \dots, n$ ,  $l(R_O(P_{\pi_j})) = \min(Y_j, M) + \min(Y_{j+1}, M)$ , and for  $j = 0, \dots, n$ ,  $l(R_N(P_{\pi_j})) = Y_{j+1} - \min(Y_{j+1}, M)$ . We assume that the left ends of the clones are uniformly distributed over the interval  $[0, N - M]$ . Therefore it can be shown that for  $j = 1, \dots, n - 1$ , the probability  $P_{Both}$  that a randomly chosen clone will fall in the region  $R_B(P_{\pi_j}, P_{\pi_{j+1}})$  is given by  $P_{Both} = \frac{M - \min(Y_{j+1}, M)}{N - M}$ , for  $j = 1, \dots, n$  the probability  $P_{Only}$  that a randomly chosen clone will fall in the region  $R_O(P_{\pi_j})$  is given by  $P_{Only} = \frac{\min(Y_j, M) + \min(Y_{j+1}, M)}{N - M}$ , and for  $j = 0, \dots, n$  the probability  $P_{None}$  that a randomly chosen clone will fall in the region  $R_N(P_{\pi_j})$  is given by  $P_{None} = \frac{Y_{j+1} - \min(Y_{j+1}, M)}{N - M}$  [3].

Let  $O_{i,j}$  be the event that the clone  $i$  will fall in the region  $R_O(P_{\pi_j})$ ;  $B_{i,j}$  the event that the clone  $i$  will fall in the region  $R_B(P_{\pi_j}, P_{\pi_{j+1}})$  and  $N_{i,j}$  the event that the clone  $i$  will fall in the region  $R_N(P_{\pi_j})$ . Then the conditional probability of observing a clonal signature  $H_i$  (i.e., the  $i$ th row in  $H$ ) given a probe ordering  $\Pi$  and an inter-probe spacing vector  $Y$  is given by

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

Given  $\Pi$ ,  $Y$  and  $O_{i,j}$ , implies that only  $h_{i,\pi_j} = 1$  and all the remaining entries in row  $H_i$  should be  $= 0$ . 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,

$$h_{i,\pi_j} = \begin{cases} 0 & \text{with probability } \eta \\ 1 & \text{with probability } (1 - \eta) \end{cases} \quad (2)$$

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

$$h_{i,\pi_k} = \begin{cases} 0 & \text{with probability } (1 - \rho) \\ 1 & \text{with probability } \rho. \end{cases} \quad (3)$$

We assume that the false positive and false negative errors at different positions along the clonal signature  $H_i$  are independent of each other. Hence  $P(H_i | \Pi, Y, O_{i,j}) = (1 - \eta)^{h_{i,\pi_j}} \cdot \eta^{(1-h_{i,\pi_j})} \cdot \rho^{(p_i-h_{i,\pi_j})} \cdot (1-\rho)^{(n-1)-(p_i-h_{i,\pi_j})}$ . Following the same argument we can show that  $P(H_i | \Pi, Y, B_{i,j}) = (1 - \eta)^{(h_{i,\pi_j}+h_{i,\pi_{j+1}})} \cdot \eta^{(2-h_{i,\pi_j}-h_{i,\pi_{j+1}})} \cdot \rho^{(p_i-h_{i,\pi_j}-h_{i,\pi_{j+1}})} \cdot (1 - \rho)^{(n-2)-(p_i-h_{i,\pi_j}-h_{i,\pi_{j+1}})}$  and  $P(H_i | \Pi, Y, N_{i,j}) = \rho^{p_i} \cdot (1 - \rho)^{(n-p_i)}$ . Hence we get,

$$\begin{aligned} P(H_i | \Pi, Y) &= \sum_{j=1}^n \left[ (1 - \eta)^{h_{i,\pi_j}} \cdot \eta^{(1-h_{i,\pi_j})} \cdot \rho^{(p_i-h_{i,\pi_j})} \cdot (1 - \rho)^{(n-1)-(p_i-h_{i,\pi_j})} \cdot \frac{\min(Y_j, M) + \min(Y_{j+1}, M)}{N - M} \right] \\ &+ \sum_{j=1}^{n-1} \left[ (1 - \eta)^{(h_{i,\pi_j}+h_{i,\pi_{j+1}})} \cdot \eta^{(2-h_{i,\pi_j}-h_{i,\pi_{j+1}})} \cdot \rho^{(p_i-h_{i,\pi_j}-h_{i,\pi_{j+1}})} \cdot \frac{M - \min(Y_{j+1}, M)}{N - M} \right] \\ &+ \sum_{j=0}^n \left[ \rho^{p_i} \cdot (1 - \rho)^{(n-p_i)} \cdot \frac{Y_{j+1} - \min(Y_{j+1}, M)}{N - M} \right] \end{aligned} \quad (4)$$

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  $P(H | \Pi, Y) = \prod_{i=1}^k P(H_i | \Pi, Y)$  which gives us

$$P(H | \Pi, Y) = \prod_{i=1}^k C_i \left\{ R_i - \sum_{j=1}^{n+1} (a_{i,\pi_j} - 1)(a_{i,\pi_{j-1}} - 1) \min(Y_j, M) \right\} \quad (5)$$

where

$$a_{i,j} = \begin{cases} \frac{\eta}{(1-\rho)} & \text{if } h_{i,j} = 0 \text{ and } j = 1, \dots, n \\ \frac{(1-\eta)}{\rho} & \text{if } h_{i,j} = 1 \text{ and } j = 1, \dots, n \\ 0 & \text{otherwise,} \end{cases} \quad (6)$$

$C_i = \frac{\rho^{p_i} (1-\rho)^{(n-p_i)}}{N-M}$ , and  $R_i = N - nM + M \sum_{j=1}^{(n-1)} a_{i,\pi_j} a_{i,\pi_{j+1}}$ .

The goal therefore is to determine  $\Pi$  and  $Y$  that maximize  $P(H | \Pi, Y)$  as given in equation (5), that is determine  $(\hat{\Pi}, \hat{Y})$  where  $(\hat{\Pi}, \hat{Y}) = \arg \max_{(\Pi, Y)} P(H | \Pi, Y)$ . Alternatively we could consider the negative log-likelihood (NLL) function  $f(\Pi, Y)$  given by

$$f(\Pi, Y) = -\ln P(H | \Pi, Y). \quad (7)$$

Since  $\ln x$  is a monotonically increasing function of  $x$  for all  $x > 0$ , it follows that  $(\hat{\Pi}, \hat{Y}) = \arg \max_{(\Pi, Y)} P(H | \Pi, Y) = \arg \min_{(\Pi, Y)} f(\Pi, Y)$ .

## 2.3 Computation of the ML Estimate

Computing the values of  $\hat{\Pi}$  and  $\hat{Y}$  involves a two stage procedure:

**Stage 1:** We first determine the optimal spacing  $\hat{Y}_{\Pi}$  for a given probe ordering  $\Pi$  i.e., determine  $\hat{Y}_{\Pi} = (\hat{Y}_1, \dots, \hat{Y}_n)$  such that for a given  $\Pi$ ,  $f(\Pi, \hat{Y}_{\Pi}) = \min_Y f(\Pi, Y) = \min_Y f_{\Pi}(Y)$ . Here the minimum is taken over all feasible solutions  $Y$  that satisfy the constraints  $Y_i \geq 0$ ;  $i = 1, \dots, n$  and  $\sum_{i=1}^n Y_i \leq N - nM$ .

**Stage 2:** We determine  $\hat{\Pi}$  for which,  $f(\hat{\Pi}, \hat{Y}_{\hat{\Pi}}) = \min_{\Pi} f(\Pi, \hat{Y}_{\Pi}) = \min_{\Pi} f_{\hat{Y}_{\Pi}}(\Pi)$ . Here the minimum is taken over all  $\Pi$  where  $\Pi$  is a permutation of  $\{1, \dots, n\}$ . The resulting values of  $\hat{\Pi}$  and  $\hat{Y}_{\hat{\Pi}}$  are termed the ML estimates (MLEs) of the true probe ordering and the inter-probe spacings, respectively.

### 2.3.1 Computation of $\hat{Y}_{\Pi}$

It can be shown that  $f_{\Pi}(Y)$  is convex in a finite number of closed regions in  $\mathcal{F} \subseteq \mathcal{R}^n$  and therefore possesses a unique local minimum which is also a global minimum [3, 10]. Consequently this minimum can be reached using continuous local search-based techniques such as the steepest descent search [7]. The steepest descent search is a simple iterative procedure which 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, or in practice, until the gradient magnitude is less than a prespecified threshold. The local downhill gradient is given by  $-\nabla f(\Pi, \hat{Y}) = -(\frac{\partial f(\Pi, Y)}{\partial Y_1}, \dots, \frac{\partial f(\Pi, Y)}{\partial Y_n})|_{Y=\hat{Y}} = (U_1, \dots, U_n)|_{Y=\hat{Y}} = U|_{Y=\hat{Y}}$ . The current value of  $\hat{Y} = \hat{Y}_{old}$  is updated by moving along the downhill gradient direction  $U$ . The new value of  $\hat{Y} = \hat{Y}_{new}$  is given by  $\hat{Y}_{new} = \hat{Y}_{old} + sU$ . The problem, therefore, is to find an optimal value of  $s$ , say  $s^*$  such that  $f(\Pi, \hat{Y} + s^*U) = \min_s f(\Pi, \hat{Y} + sU)$ . Having obtained the value of  $s^*$ , the new inter-probe spacings are given by  $\hat{Y}_{new} = \hat{Y}_{old} + s^*U$ .

To determine an optimal value of  $s = s^*$  we exploit the convexity of  $f_{\Pi}(Y)$  which implies that the local optimum for  $s$  is also a global optimum. Using the constraints that the spacings are non-negative, the clones and probes are of fixed length and the total length of the chromosome is fixed, we compute the upper and lower bounds on the values of  $s$  and use the bisection method to find the optimal value of  $s = s^*$ . If any of the boundary conditions (represented as hyperplanes) on the  $Y_i$ 's for  $i = 1, \dots, n$  are violated, the gradient vector  $U$  is projected onto the admissible region which is represented as the intersection of the  $k$  hyperplanes corresponding to the  $k$  violated constraints. The minimization procedure then proceeds along the projected gradient direction  $U_{proj}$  instead of  $U$ . In the limiting case when  $k = n$ , the minimization procedure has reached an extremal vertex of the admissible region and  $U_{proj} = 0$ . 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.

In this paper, we have used the conjugate gradient descent (CGD) search instead of the steepest descent search since the former is known to be one of the fastest in the class of gradient descent-based optimization methods [7]. The CGD search 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 direction, 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 [7]. Unlike the steepest descent algorithm, the CGD algorithm

guarantees convergence to a local minimum within  $n$  steps.

### 2.3.2 Computation of $\hat{\Pi}$

Determining the optimal clone ordering  $\hat{\Pi}$ , entails a combinatorial search through the discrete space of all possible permutations of  $\{1, \dots, n\}$ . The problem of coming up with such an optimal ordering is isomorphic to the classical NP-complete *Traveling Salesman Problem* (TSP) for which no polynomial-time algorithm for determining the optimal solution is known [12]. One could use a simulated Monte Carlo search method such as Simulated Annealing (SA) [6] or the Large Step Markov Chain (LSMC) [13] or an evolutionary search (i.e., population-based Monte Carlo search) method [14], all of which are known to be robust in the presence of local optima in the solution space and give near-optimal solutions in average polynomial time. Our previous work in using SA and LSMC indicated that whereas these methods were robust to the presence of local optima, their parallel versions were not scalable with respect to speedup and efficiency of parallelism with an increasing number of processors [4]. Consequently, we decided to investigate evolutionary search methods and their parallel implementation on a network of shared-memory symmetric multiprocessors (SMPs).

A typical evolutionary method begins with an initial ensemble or population of candidate solutions and iterates through a number of generations before reaching a locally optimal solution. In each iteration or generation, the solutions in the current population are subject to evolutionary operators that mimic their biological counterparts [14]. The Genetic Algorithm (GA) is one of the most widely used evolutionary methods. A typical GA uses the selection, crossover and mutation operations on the candidate solutions in the current population to generate solutions for the following generation.

The selection operator uses a *roulette-wheel* procedure to select candidate solutions with probability in direct proportion to their fitness values [14]. The fitness function is the negative of the NLL objective function in equation (7) so that solutions with lower objective function values have higher fitness values and conversely. During the crossover operation the solutions selected by the roulette-wheel procedure are treated as parental chromosomes and child chromosomes are generated by exchanges of parental chromosomal segments. This mimics the phenomenon of recombination in biological chromosomes and enables large-scale exploration of the search space [14]. In our case, we used an improvised version of the heuristic crossover operator originally proposed by Jog *et al.* [8] in the context of the GA for the TSP. In a typical GA for the TSP, the mutation operator is a random local perturbation and is typically implemented using the 2-opt perturbation [12] where the ordering within a randomly chosen block of probes in the current probe ordering is reversed. The GA is deemed to have converged when the best solution in the population has not improved within a certain number of successive generations.

In the absence of a hill-climbing mechanism, the GA exhibits a slow convergence rate. The incorporation of *deterministic* hill-climbing into the GA typically results in premature convergence to a local optimum [2]. Our previous experience with the GA has shown that incorporation of a *stochastic* hill-climbing search mechanism greatly improves the asymptotic convergence of the GA to a near-globally optimal solution [2]. As a consequence, the GA was enhanced with the incorporation of a stochastic hill-climbing search similar to that of the LSMC algorithm [13]. The stochastic hill-climbing search in the LSMC algorithm incorporates an exhaustive local search using the 2-opt perturbation coupled with the Metropolis or Boltzmann decision function [4]. The GA-LSMC hybrid algorithm is outlined in Figure 1. Note that the members of the population in each generation are locally optimal solutions under the 2-opt perturbation. This is ensured by a

### The GA-LSMC Hybrid Algorithm with Stochastic Population Replacement:

1. Create an initial population of locally optimal solutions using the *double-bridge* perturbation and an exhaustive local 2-opt search; (This ensures that all the members in the initial population are locally optimal solutions)
2. While not converged do
  - (a) Select two parents using the roulette wheel selection procedure;
  - (b) Apply the heuristic crossover to the selected parents to create an offspring  $S$ ;
  - (c) Perform exhaustive local 2-opt search on  $S$  to yield a new locally optimum solution  $S'$ ;
  - (d) With probability  $p_m$  perform a mutation on  $S'$  using a double-bridge perturbation followed by exhaustive local 2-opt search to yield a new locally optimum solution  $S^*$ ;
  - (e) Evaluate the NLL objective function at  $S^*$  (if mutation is performed) or at  $S'$  (if mutation is not performed) using conjugate gradient descent search;
  - (f) Compute  $\Delta f = f(S^*) - f(P)$  (if mutation is performed) or  $\Delta f = f(S') - f(P)$  (if mutation is not performed) where  $P$  is the less fit of the two parents;
  - (g) Replace  $P$  with  $S^*$  (if mutation is performed) or  $S'$  (if mutation is not performed) with probability  $p_r$  computed using the Boltzmann function  $p_r = \frac{1}{1 + \exp(\frac{\Delta f}{T})}$ ;
  - (h) Update  $p_m$ ;
  - (i) Update temperature using the annealing function  $T = A(T)$ ;
  - (j) Check for convergence;
3. Output the best solution in the population as the final solution;

Figure 1: Outline of the GA-LSMC hybrid algorithm with stochastic replacement

mutation operator which is a combination of the non-local *double-bridge* perturbation [13] and an exhaustive local search using the 2-opt perturbation.

## 3 Parallel Computation of the ML Estimator

We propose a two-tier parallel computation of the ML estimator corresponding to the two stages of optimization.

**Level 1:** Parallel computation of the optimal inter-probe spacing  $\hat{Y}_{\Pi}$  for a given probe ordering  $\Pi$  that minimizes  $f(\Pi, \hat{Y}_{\Pi})$ . 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  $\hat{\Pi}$  for which  $f(\hat{\Pi}, \hat{Y}_{\hat{\Pi}})$  is minimum. This entails parallelization of the GA-LSMC hybrid algorithm for optimization in the *discrete* domain.

The parallel algorithms were implemented on a cluster of SMPs using a combination of MPI and multi-threaded programming.

### 3.1 Parallel Evolutionary Search

Our approach to parallelizing the GA is based on partitioning the population amongst the available processors [5]. Each processor is responsible for searching for the best solution within its subpopulation. This is tantamount to performing multiple concurrent searches within the search

space [5]. The parallel GA (pGA) was implemented using MPI on a distributed-memory platform comprising of a network of SMPs using the master-slave model. The master process runs on one of the SMPs within the SMP cluster. The master process reads in the problem data, creates the initial population, spawns slave processes on all the SMPs (including its own) and divides the initial population amongst the slave processes. Each slave process runs a serial version of the GA-LSMC hybrid algorithm (Figure 1) on its subpopulation concurrently with the other slave processes. The slave processes periodically send the solutions within their subpopulation to the master process. The master process on receipt of the solutions from all the slave processes, checks for convergence and, mixes the solutions at random and redistributes the population amongst the slave processes. This periodic mixing and redistribution of the population prevents a slave process from premature convergence to a local optimum after having exhausted all the genetic variation within its subpopulation. The master process deems the pGA to have converged if the best solution in the overall population has not changed over a certain number of successive generations.

### 3.2 Parallel Gradient Descent Search

Due to its inherent sequential nature, we deemed data parallelism to be appropriate for the parallel CGD algorithm. The  $Y$  and  $U$  vectors are distributed amongst the different processors within an single SMP and each processor performs the gradient vector computation and updates to the inter-probe spacing vector using its local subvectors  $Y_{loc}$  and  $U_{loc}$  concurrently with the other processors within the SMP. Here,  $|Y_{loc}| = |Y|/N_P$  and  $|U_{loc}| = |U|/N_P$  where  $N_P$  is the number of processors within each SMP. A multi-threaded programming approach [1] was used with a single thread running on a single processor within the SMP. Since the individual subvectors have to be periodically distributed amongst the processors and also periodically gathered to compute a global value for  $s$  during the bisection procedure, the threads have to be periodically synchronized using a barrier. Updates to global values in shared memory (such as the global sum of the vector components) are controlled using mutex (mutual exclusion) locks.

### 3.3 A Two-tier Parallelization of the ML Estimator

In order to ensure a scalable implementation, two tiers of parallelism were incorporated in the computation of the ML estimator. The finer or lower level of parallelism pertains to the computation of  $\hat{Y}$  for a given probe ordering  $\Pi$  using the parallel multi-threaded CGD algorithm for continuous optimization. The coarser or upper level of parallelization pertains to the computation of  $\hat{\Pi}$  using the pGA for discrete optimization. The multi-threaded CGD algorithm is embedded within each of the pGAs and, as such, the parallelization of the CGD algorithm at the finer level is transparent to the pGA at the coarser level. When the parallel CGD procedure is invoked from within the master or slave pGA process, a new set of slave CGD processes (i.e. threads) is spawned on the available processors (within an SMP), whereas the master CGD process (i.e. thread) runs on the same processor as the pGA process (master or slave). The master and slave CGD processes (i.e. threads) cooperate to evaluate and minimize the value of  $f(\Pi, \hat{Y}_{\Pi})$ . Once  $f(\Pi, \hat{Y}_{\Pi})$  is minimized, the slave CGD processes (i.e. threads) terminate and the corresponding processors within an SMP are available for future computation. The two-tier parallelism approach induces a logical tree-shaped interconnection network on the processors within the SMP cluster.

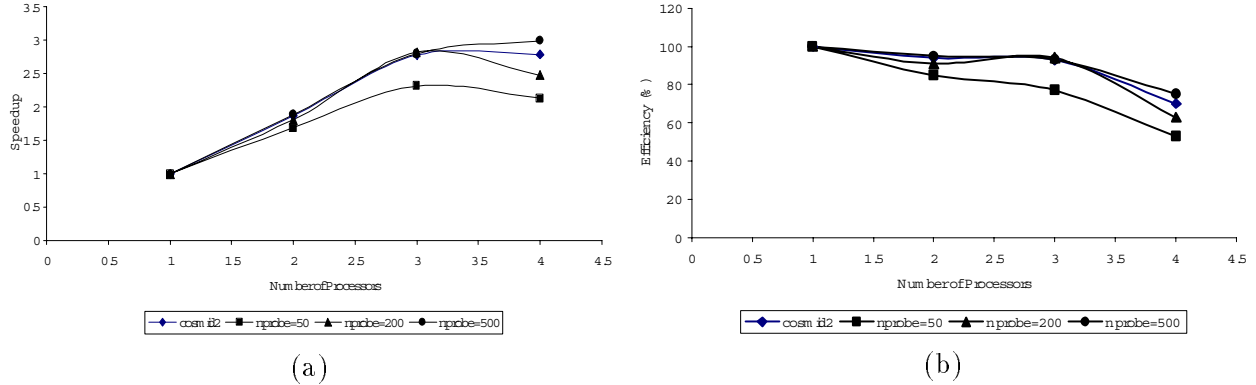


Figure 2: Parallel (multithreaded) CGD algorithm: (a) Speedup curves, (b) Efficiency curves

## 4 Experimental Results

The parallel algorithms were implemented on a dedicated cluster comprising of 8 nodes where each node is a shared-memory symmetric multiprocessor (SMP) running SUN Solaris-x86. Each SMP comprises of 4 700MHz Pentium III Xeon processors with 1 MB cache per processor and 1GB of shared memory. The programs were tested with simulated clone-probe hybridization data [3] as well as real data from *cosmid2* ( $n = 109, k = 2046$ ) and *cosmid3* ( $n = 111, k = 1937$ ) of the fungal genome *Neurospora crassa* being currently mapped in the Department of Genetics at the University of Georgia.

The parallel CGD algorithm was tested on simulated chromosomal data sets with a varying number of probes and clones  $(n, k) = (50, 300), (200, 1300)$  and  $(500, 3250)$  and on real data from *cosmid2*. Figure 2(a) shows the resulting speedup curves and Figure 2(b) the resulting efficiency curves. These results are in conformity with our expectations since the inter-thread synchronization overhead and the wait times tend to increasingly dominate the overall execution time with an increasing number of processors for a given value of  $n$ . The payoff in the parallelization of the CGD algorithm is better realized for larger values of  $n$  (i.e., larger problem sizes).

The parallel multithreaded exhaustive local search algorithm was also tested on the simulated data set. The speedup and efficiency curves for the parallel multithreaded exhaustive local search algorithm are shown in Figure 3. As can be seen, the payoff in parallelization is better realized for larger values of  $n$  (i.e., larger problem sizes). For a given problem size, the efficiency is seen to decrease with an increasing number of processors  $N_P$  within the SMP, but not appreciably. This can be attributed to the fact that although there is a certain amount of overhead involved in creation of multiple slave threads and binding them to distinct processors, there is very infrequent synchronization amongst the slave threads or between the master thread and the slave threads. Since each slave thread performs an independent search of the space of 2-opt perturbations, the only synchronization needed is at the beginning and end of the search process. This is in contrast to the parallel multithreaded CGD search algorithm where the synchronization amongst the slave threads or between the master thread and the slave threads is much more frequent.

The pGA was implemented with the following parameters: the population size  $N_{pop}$  was chosen to be 40, the initial temperature  $T_{init}$  was chosen to be 1, the annealing factor  $\alpha$  in the geometric annealing schedule  $T_{next} = \alpha \cdot T_{prev}$  was chosen to be 0.95, the maximum number of trials *max\_trials* before the population was mixed was chosen to be the population size  $N_{pop}$  and the convergence criterion used was the fact that no member in the population was replaced for two successive

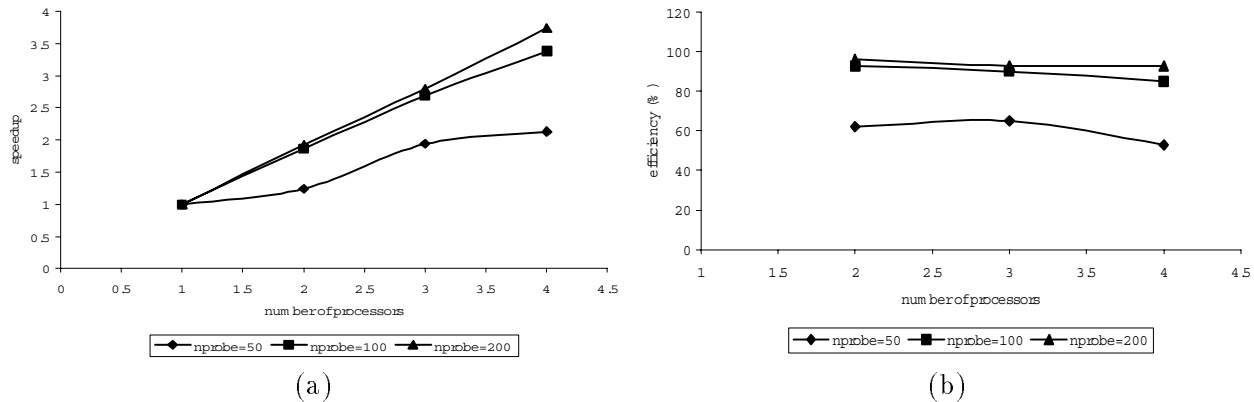


Figure 3: Parallel (multithreaded) exhaustive local search algorithm using the 2-opt heuristic: (a) Speedup curves, (b) Efficiency curves

Table 1: Comparison of the serial SA, LSMC and GA-LSMC algorithms

Data Set	GA-LSMC			LSMC			SA		
	T (sec)	NLL	C	T (sec)	NLL	C	T (sec)	NLL	C
$n = 50, k = 300$	3325	1548.39	4	10746	1624.09	12	15076	1665.37	12
$n = 100, k = 650$	10919	4262.8	9	7459	4297.5	14	6265	4288.64	13
$n = 200, k = 1300$	181311	11159.48	9	105893	11515.13	24	31013	11574.75	27
<i>cosmid2</i>	27962	12731.37	-	34704	12757.55	-	108499	12949.99	-
<i>cosmid3</i>	14922	12584.62	-	30183	12501.88	-	45533	13212.85	-

T: Execution time, NLL: NLL fuction value, C: Number of contigs recovered

generations (i.e., the population remained the same for two successive generations). Note that this is a stricter convergence criterion than the one that requires only the best solution in the population to be unchanged over a certain number of successive generations. The mutation probability  $p_m$  was set dynamically as follows: for a population replacement rate greater than 70% of  $N_{pop}$ ,  $p_m$  was set to 0, for a population replacement rate between 30% and 70% of  $N_{pop}$ ,  $p_m$  was set to 0.1, and for a population replacement rate less than 70% of  $N_{pop}$ ,  $p_m$  was set to 0.2. Thus, the mutation probability is kept low when the population replacement rate is sufficiently high. The crossover operation is the primary mechanism for exploration of the search space and for maintaining genetic variation in the population in this case. When the genetic variation in the population has depleted after repeated selection and crossover operations, resulting in a low population replacement rate, the mutation probability is gradually raised to introduce more genetic variation into the population.

The results of the serial GA-LSMC hybrid algorithm were compared with those of the serial SA and serial LSMC algorithm (Table 1). The serial GA-LSMC hybrid algorithm was seen to consistently yield lower NLL values compared to the SA and LSMC algorithms on both, artificial and real data. The only exception is the real data set *cosmid3* where the GA-LSMC hybrid algorithm yielded a slightly higher NLL function value (less than 1% difference) than the LSMC algorithm but with much shorter execution time (less than half). Table 1 also shows the number of probe suborderings (i.e., contigs) recovered by the GA-LSMC, SA and LSMC algorithms on the synthetic data sets. In an ideal case, one should be able to recover the true probe ordering as a single contig. In reality, this is unlikely due to the presence of hybridization errors. In a realistic scenario, the physical mapping algorithm would be expected to recover probe suborderings

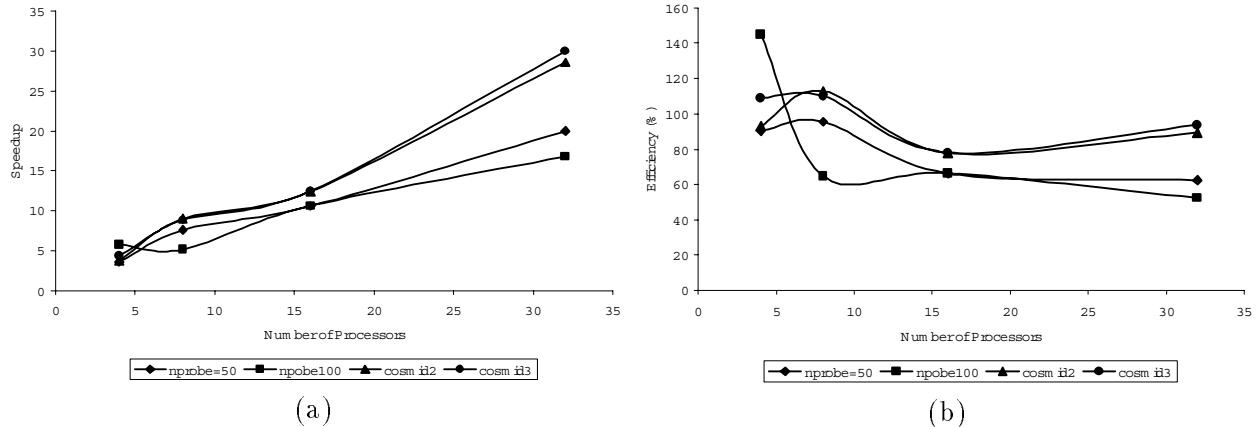


Figure 4: Parallel GA with stochastic replacement: (a) Speedup curves, (b) Efficiency curves

that could then be manually manipulated (via translation and probe order reversal) to yield the final probe order. The fewer and longer these probe suborderings or contigs, the less intensive the subsequent manual editing to recover the desired probe ordering. The GA-LSMC algorithm was seen to consistently yield fewer and longer contigs than the SA and LSMC algorithms suggesting that the GA-LSMC algorithm is capable of yielding solutions of higher quality.

The performance of the pGA is summarized in Figure 4. The pGA was seen to exhibit superlinear speedup in some instances. This is partly due to the fact that with a larger number of processors, the population per processor decreases to the point where it can reside entirely in cache. For a small number of processors, this caching effect can overcome the inter-processor communication and synchronization overhead resulting in super-linear speedup. Also, due to the inherently stochastic nature of the selection, crossover and mutation operations in the GA, the manner in which the search space is traversed by the serial GA and the parallel GA could be entirely different. The difference in the manner of the search tree traversal has been known to cause instances of superlinear speedup in the case of other well known combinatorial search algorithms such as branch-and-bound [11]. Also, the efficiency values are observed to be higher for larger problem instances (larger values of  $n$  and  $k$ ) for a given value of  $N_{proc}$  (the total number of processors). The efficiency values also exhibit an overall declining trend for increasing values of  $N_{proc}$  for a given problem instance implying the dominance of the inter-processor communication and synchronization overhead in the overall execution time. These observations are in conformity with the general expectations regarding the performance of the pGA.

## 5 Conclusions and Future Directions

In this paper we presented a maximum likelihood (ML) estimation-based approach to physical map reconstruction under a probabilistic model of hybridization errors consisting of false positives and false negatives. The ML estimate reconstructs the optimal probe ordering and optimal inter-probe spacings when used in conjunction with the sampling-without-replacement experimental protocol. The estimation procedure was shown to entail continuous optimization for determining the optimal inter-probe spacings for a given probe ordering and combinatorial optimization for determining the optimal probe ordering. A two-tier parallelization strategy was proposed wherein the CGD search algorithm for continuous optimization is parallelized at the lower level and the GA-LSMC hybrid algorithm for combinatorial optimization is simultaneously parallelized at the higher level. The

parallel ML estimation algorithm was shown to be amenable to efficient implementation on a network of SMPs where the CGD search is parallelized on a single SMP using shared-memory, multi-threaded programming whereas the GA-LSMC algorithm is parallelized on the SMP network using the distributed-memory, message-passing-based programming paradigm within the MPI environment.

Future research will investigate extensions of the maximum likelihood function that also encapsulate errors due to repeat DNA sequences in addition to false positives and false negatives. The current implementation of the ML estimator is targeted towards a homogeneous platform such as a network of identical SMPs. Future research will explore and address issues that deal with the parallelization of the ML estimator on a heterogeneous platform such as a network of SMPs that differ in processing speeds and memory capacity, since that is a scenario that is more likely to be encountered in the real world.

**Acknowledgments:** This research was supported in part by an NRICGP grant by the US Department of Agriculture and a Research Instrumentation grant by the National Science Foundation.

## References

- [1] G. Andrews, *Foundations of Multithreaded, Parallel, and Distributed Programming*, Addison Wesley Pub. Co., Reading, MA, 2000.
- [2] S.M. Bhandarkar and H. Zhang, Image Segmentation using Evolutionary Computation, *IEEE Trans. Evolutionary Computation*, Vol. 3, No. 1, pp. 1–21, April 1999.
- [3] S.M. Bhandarkar, S.A. Machaka, S.S. Shete and R.N. Kota, Parallel Computation of a Maximum Likelihood Estimator of a Physical Map, *Genetics*, special issue on Computational Biology, Vol. 157, No. 3, pp. 1021–1043, March 2001.
- [4] S.M. Bhandarkar, J. Huang and J. Arnold, Parallel Monte Carlo Methods for Physical Mapping of Chromosomes, *Proc. IEEE Bioinformatics Conference*, Stanford University, Palo Alto, CA, pp. 64–75, August 2002.
- [5] E. Cantu-Paz, *Efficient and Accurate Parallel Genetic Algorithms*, Kluwer Academic Publishers, Boston, MA, November 2000.
- [6] S. Geman and D. Geman, Stochastic Relaxation, Gibbs Distribution and the Bayesian Restoration of Images, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 6, pp. 721–741, 1984.
- [7] M. Hestenes and E. Stiefel, Methods of Conjugate Gradients for Solving Linear Systems. *Journal of Research of the National Bureau of Standards*, Vol. 49, pp. 409–436, 1980.
- [8] P. Jog, J.Y. Suh, and D. Van Gucht, The Effects of Population Size Heuristic Crossover and Local Improvement on a Genetic Algorithm for the Traveling Salesman Problem, *Proc. Intl. Conf. Genetic Algorithms*, Fairfax, VA, pp. 110–115, June 1989.
- [9] J.D. Kececioglu and E.W. Myers, Combinatorial Algorithms for DNA Sequence Assembly, *Algorithmica*, Vol. 13, pp. 7–51, 1995.
- [10] J.D. Kececioglu, S.S. Shete and J. Arnold, Reconstructing Distances in Physical Maps of Chromosomes With Nonoverlapping Probes, *Proc. 4th ACM Conf. Comp. Mol. Biol. (RECOMB)*, Tokyo, Japan, pp. 183–192, April, 2000.
- [11] T.H. Lai and S. Sahni, Anomalies in parallel branch and bound algorithms, *Comm. ACM*, Vol. 27, No. 6, pp. 594–602, June 1984.
- [12] S. Lin and B. Kernighan, An Effective Heuristic Search Algorithm for the Traveling Salesman Problem, *Operations Research*, Vol. 21, pp. 498–516, 1973.
- [13] O. Martin, S.W. Otto and E.W. Felten, Large-Step Markov Chains for the Traveling Salesman Problem, *Complex Systems*, Vol. 5, No. 3, pp. 299–326, 1991.
- [14] M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA, 1996.
- [15] P. Pacheco, *Parallel Programming with MPI*, Morgan Kaufmann Publishers, San Francisco, CA, 1996.