1

An Information Theoretic Approach to Genome Reconstruction

Suchendra Bhandarkar
Department of Computer Science, The University of Georgia

Jinling Huang
Center for Tropical and Emerging Global Diseases, The University of Georgia

Jonathan Arnold
Department of Genetics, The University of Georgia

1.1 Introduction

Creating maps of entire chromosomes, which could then be used to reconstruct the chromosome's DNA sequence, has been one of the fundamental problems in genetics right from its very inception [Stu13]. These maps are central to the understanding of the structure of genes, their function, their transmission and their evolution. 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 million base pairs (Mb). Lander and Green [LG87] pioneered the use of computational techniques for the assembly of genetic maps with many markers. 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 thousand base pairs (Kb). 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 molecular manipulation of genes of interest.

1.1.1 The Physical Mapping Protocol

The physical mapping protocol essentially determines the nature of clonal data and the probe selection procedure. The physical mapping protocol used in our work is the one based on *sampling without replacement* [FTA92]. This protocol is simple, adaptable and relatively inexpensive and has been used successfully in physical mapping projects of several fungal genomes under the Fungal Genome Initiative at the University of Georgia [Arn97].

The protocol that generates the probe set $\mathcal{P}$ and the clone set $\mathcal{C}$ is an iterative procedure which 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 attaches (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 $\mathcal{C}^0$ that covers the length of the chromosome.

The clone-probe overlap pattern is represented in the form of a binary hybridization matrix H where H_{ij} denotes the hybridization of the ith clone $\in \mathcal{C}$ to the jth probe $\in \mathcal{P}$. $H_{ij} = 1$ if the ith clone $\in \mathcal{C}$ hybridizes to the jth probe $\in \mathcal{P}$ and $H_{ij} = 0$ otherwise. 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 [KM95] 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. 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 paper, we confine ourselves to errors in the form of false positives and false negatives. Since the clones (and probes) in the mapping projects at the University of Georgia that use the aforementioned protocol are generated using cosmids which makes them sufficiently small (around 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.

1.1.2   Computation of a Physical Map

Several techniques exist for computation 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 [FAW95], mapping by unique probes [FAZ94, GI95, JM97], mapping by unique endprobes [CJK$^+$97], mapping using restriction fragments [FJK$^+$97, JK97], mapping using radiation-hybrid data [BDC97, SKSL97] and optical mapping [KS98, LDW98, MP97]. Likewise, several computation techniques based on deterministic optimization and stochastic optimization have been reported in the literature in the context of physical mapping. Examples of stochastic optimization algorithms include simulated annealing [FAZ94, FAW95, CAT92, MGM$^+$93], and the random cost algorithm [WPG$^+$94] whereas those of deterministic optimization algorithms include linear programming [JM97], integer programming [CJK$^+$97], integer linear programming with polyhedral combinatorics [CK99] and semidefinite programming [CS95]. Various statistical analyses of the aforementioned physical mapping techniques have also been reported in the literature [RAW91, Bal94, LW88, NS94, DWP97, XCP$^+$96, ZM93].

In this chapter we describe a physical mapping approach based on the information theoretic concept of maximum likelihood reconstruction of signals that have been corrupted by noise when transmitted through a communications channel. We model the chromosome as the original signal, the hybridization experiments as the process of transmission through a communications channel, the hybridization errors as the noise introduced by the communications channel, the hybridization matrix as the observed corrupted signal at the receiving end of the communications channel and the desired physical map as the reconstructed signal. In particular, we describe the maximum likelihood (ML) estimation-based approach to physical map construction proposed in [SBK01, KSA00] which determines the probe ordering and inter-probe spacings that maximize the probability of occurrence of the experimentally observed hybridization matrix H 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. The problem of determining the optimal probe ordering is intractable and can be shown to be isomorphic to the classical NP-hard Traveling Salesman Problem (TSP) [GJ79]. Moreover, the ML objective function is non-linear in the probe ordering thus rendering the classical linear programming or integer linear programming techniques inapplicable. However, for a given probe ordering, determining the optimal inter-probe spacings is shown to be a tractable problem that is solvable using gradient descent-based search techniques.

In this chapter we present three stochastic combinatorial optimization algorithms for computation of the optimal probe ordering based on simulated annealing (SA), large step Markov chains (LSMC) and the genetic algorithm (GA). The computation of the optimal inter-probe spacings for a specified probe ordering is shown to be best achieved by the conjugate gradient descent algorithm. We propose a two-tier parallelization strategy for efficient implementation of the ML estimation-based physical mapping algorithm. The upper level represents parallel discrete optimization using the aforementioned stochastic combinatorial optimization algorithms whereas the lower level comprises of parallel conjugate gradient descent search. The resulting parallel algorithms are implemented on a cluster of shared-memory symmetric multiprocessors (SMPs). The conjugate gradient descent search algorithm is parallelized using shared-memory multithreaded programming on a single SMP whereas the stochastic combinatorial optimization algorithm is implemented on the SMP cluster using the distributed-memory Message Passing Interface (MPI) en-

vironment [Pac96]. Convergence, speedup and scalability characteristics of the parallel algorithms are compared, analyzed and discussed.

1.2 The Maximum Likelihood Reconstruction of a Physical Map

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 protocol described in Section 1.1.1, 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 inter-probe 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 and the left end of probe P_{π_i} (where $2 \leq i \leq n$). The spacing between the right end of probe P_{π_n} and the right end of the chromosome is given by $Y_{n+1} = N - nM - \sum_{i=1}^{n} Y_i$ where N is length of the chromosome and M is the length of each clone/probe. Note that the protocol described in Section 1.1.1 requires that all probes and clones be of the same length.

The problem as stated above is ill-posed as defined by Hadamard [Had23] since the underlying constraints do not imply a unique solution. In the absence of errors, any probe ordering $(\pi_1, \pi_2, \ldots, \pi_n)$ with inter-probe spacings $Y = (Y_1, Y_2, \ldots, Y_n)$ that satisfies the constraint $N \geq nM + \sum_{i=1}^{n} Y_i$ is a feasible solution. In the presence of errors the problem is formulated as one of determining a probe ordering and the inter-probe spacings that maximize the likelihood of the observed hybridization matrix H given predefined probabilities for false positives and false negatives.

1.2.1 Mathematical Notation

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

N : Length of the chromosome (in base pairs),

M : Length of a clone/probe (in base pairs),

n : Number of probes,

k : Number of clones,

ρ : Probability of false positive,

η : 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 : ith row of the hybridization matrix (also termed as the binary hybridzation signature of the ith clone)

$\Pi = (\pi_1, \ldots, \pi_n)$: permutation of $\{1, 2, \ldots, n\}$ which denotes the probe labels in the ordering when scanned from left to right along the chromosome,

$p_i = \sum_{j=1}^{n} h_{i,j}$: number of 1's in H_i,

$P = \sum_{i=1}^{k} p_i$: total number of 1's in H,

$Y = (Y_1, Y_2, \ldots, Y_n)$: vector of inter-probe spacings where Y_i is the spacing between the right end of $P_{\pi_{i-1}}$ and the left end of P_{π_i} ($2 \leq i \leq n$), 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 inter-probe spacings $Y = \{Y_1, \ldots, Y_n\}$ such that $Y_i \geq 0$, $1 \leq i \leq n$ and $N - nM - \sum_{i=1}^{n} Y_i \geq 0$.

Under the assumptions that (i) the false positive and false negative errors at different positions along the clonal hybridization signature H_i are independent of each other, and (b) the clones $\in \mathcal{C}$ are independently distributed along the chromosome i.e., each row of H is independent of the other rows, the probability of observing a hybridization matrix H for a given probe ordering Π and inter-probe spacing vector Y is given by:

$$P(H \mid \Pi, Y) = \prod_{i=1}^{k} C_i \left\{ R_i - \sum_{j=1}^{n+1} (a_{i,\pi_j} - 1)(a_{i,\pi_{j-1}} - 1) \min(Y_j, M) \right\} \qquad (1.1)$$

where

$$a_{i,j} = \begin{cases} \frac{\eta}{(1-\rho)} & \text{if } h_{i,j} = 0 \text{ and } j = 1, \ldots, n \\ \frac{(1-\eta)}{\rho} & \text{if } h_{i,j} = 1 \text{ and } j = 1, \ldots, n \\ 0 & \text{otherwise,} \end{cases} \qquad (1.2)$$

$$C_i = \frac{\rho^{p_i}(1-\rho)^{(n-p_i)}}{N - M}, \qquad (1.3)$$

and

$$R_i = N - nM + M \sum_{j=1}^{n-1} a_{i,\pi_j} a_{i,\pi_{j+1}}. \qquad (1.4)$$

The detailed derivation of equation (1.1) can be found in [SBK01].

The goal therefore is to determine Π and Y that maximize $P(H \mid \Pi, Y)$ as given in equation (1.1), that is determine $(\hat{\Pi}, \hat{Y})$ where

$$(\hat{\Pi}, \hat{Y}) = \arg \max_{(\Pi, Y)} P(H \mid \Pi, Y) \qquad (1.5)$$

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

$$\begin{aligned} f(\Pi, Y) &= -\ln P(H \mid \Pi, Y) \\ &= C - \sum_{i=1}^{k} \ln \left\{ R_i - \sum_{j=1}^{n+1} (a_{i,\pi_j} - 1)(a_{i,\pi_{j-1}} - 1) \min(Y_j, M) \right\} \end{aligned} \qquad (1.6)$$

where C is a constant given by

$$C = k \ln(N - M) - P \ln \frac{\rho}{(1-\rho)} - nk \ln(1-\rho) \qquad (1.7)$$

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

$$(\hat{\Pi}, \hat{Y}) = \arg \max_{(\Pi, Y)} P(H \mid \Pi, Y) = \arg \min_{(\Pi, Y)} f(\Pi, Y) \qquad (1.8)$$

Let $\Pi^R = (\pi_n, \ldots, \pi_1)$. It is easy to verify that

$$P(H \mid \Pi, Y) = P(H \mid \Pi^R, Y) \qquad (1.9)$$

This means that the likelihood as a function of Π is unique up to reversals which implies that it is possible to recover ordering of the probes uniquely only up to reversals.

1.3   Computation of the Maximum Likelihood Estimate

Computing the values of $\hat{\Pi}$ and $\hat{Y}$ (equation (1.8)) involves a two stage procedure:

Stage 1: We 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,

$$f(\Pi, \hat{Y}_\Pi) = \min_Y f(\Pi, Y) = \min_Y f_\Pi(Y) \tag{1.10}$$

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

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

$$f(\hat{\Pi}, \hat{Y}_{\hat{\Pi}}) = \min_\Pi f(\Pi, \hat{Y}_\Pi) = \min_\Pi f_{\hat{Y}_\Pi}(\Pi) \tag{1.11}$$

Here the minimum is taken over all Π where Π is a permutation of $\{1, \ldots, n\}$.

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 $h : \mathcal{D} \mapsto \mathcal{R}$ defined on a convex set $\mathcal{D}$ is deemed convex if for all points $\alpha p + (1 - \alpha)q \in \mathcal{D}$ where $0 \leq \alpha \leq 1$, $h(\alpha p + (1 - \alpha)q) \leq \alpha h(p) + (1 - \alpha)h(q)$. 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. It can be shown that $f_\Pi(Y)$ is convex in every good convex region $\mathcal{D}$ and therefore possesses a unique local minimum which is also a global minimum [SBK01, KSA00]. Consequently this minimum can be reached using continuous local search-based techniques such as gradient descent (i.e., steepest descent) search or conjugate gradient descent search [Dor80].

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

$$\mathcal{F}_{a,b} \triangleq \{Y \in \mathcal{F} : aY_1 \leq aM; Y_i \leq M, 2 \leq i \leq N; bY_{n+1} \leq bM\} \tag{1.12}$$

Each of these regions is convex since they result from the intersection of half spaces. Also, it can be shown that since the derivative of $f_\Pi(Y)$ is defined in the interior of each subregion, each subregion is good [SBK01]. Note that we can define the derivative on the boundary of each subregion $\mathcal{F}_{a,b}$, $a, b \in \{-1, +1\}$, based on 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 $f_\Pi(Y)$ [KSA00].

Gradient Descent Search

We illustrate the computation of $\hat{Y}_\Pi$ in each of the subregions $\mathcal{F}_{a,b}$ using the steepest descent (SD) search technique. The SD search technique 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. The point at which the gradient vanishes is considered to be the desired local minimum. In practice, the SD search procedure is halted when the magnitude of the gradient is less than a prespecified threshold.

The initial value of $Y = (Y_1, \ldots, Y_n)$ in each of the subregions $\mathcal{F}_{a,b}$ can be determined by assigning $\frac{N-nM}{n+1}$ to each of Y_i's i.e., distributing the spacings equally. Having obtained

the starting value for $\hat{Y}$, we compute the gradient of the negative log likelihood function. The negative of the local gradient (i.e. the downhill gradient) represents the direction along which the function decreases the most rapidly. The local downhill gradient of $f_\Pi(Y)$ is given by

$$
\begin{aligned}
-\nabla f(\Pi, \hat{Y}) &= -(\frac{\partial f(\Pi, Y)}{\partial Y_1}, \ldots, \frac{\partial f(\Pi, Y)}{\partial Y_n}) \mid_{Y=\hat{Y}} \\
&= (U_1, \ldots, U_n) \mid_{Y=\hat{Y}},
\end{aligned}
\tag{1.13}
$$

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

$$
\hat{Y}_{new} = \hat{Y}_{old} + sU.
\tag{1.14}
$$

We attempt to find an optimal value of s, say s^* such that

$$
f(\Pi, \hat{Y} + s^*U) = \min_s f(\Pi, \hat{Y} + sU).
\tag{1.15}
$$

Having obtained the value of s^*, then the new inter-probe spacings are given by

$$
\hat{Y}_{new} = \hat{Y}_{old} + s^*U.
\tag{1.16}
$$

Our specific problem is that of constrained minimization of $f_\Pi(Y)$ in a good convex region $\mathcal{F}_{a,b}$, $a, b \in \{-1, +1\}$. The downhill gradient U at a given point Y on the boundary of $\mathcal{F}_{a,b}$ may be outside the region. In this case, we need to proceed along a direction U' that is directed inside $\mathcal{F}_{a,b}$ along which $f_\Pi(Y)$ decreases, albeit at a slower rate. Since the boundaries of $\mathcal{F}_{a,b}$ where $a, b \in \{-1, +1\}$ are hyperplanes, we determine at boundary point Y all the boundary constraints (hyperplanes) $\Gamma_1, \Gamma_2, \ldots, \Gamma_r$ that are violated by the downhill gradient U. Let $\mathbf{T}_i(U)$ denote the projection of U on Γ_i. Then the resulting direction U' is given by successively projecting U on each of the violating hyperplanes as follows

$$
U' = \mathbf{T}_r(\mathbf{T}_{r-1}(\ldots (\mathbf{T}_1(U)) \ldots)
\tag{1.17}
$$

If $U' = 0$ then the current point Y is the local minimum of $f_\Pi(Y)$ in $\mathcal{F}_{a,b}$, $a, b \in \{-1, +1\}$. If $U' \neq 0$ then the non-violating hyperplanes are used to determine upper and lower bounds on the value of s denoted by $[s_{low}, s_{high}]$. Since the function $f_\Pi(Y)$ is convex with respect to s in $\mathcal{F}_{a,b}$ where $a, b \in \{-1, +1\}$, the bisection method [PFTV88] can be used to determine s^*. The SD search procedure is terminated when either U or U' vanishes depending on which situation is encountered first.

One of the problems with the SD search is that its convergence rate is sensitive to the starting point and the shape of the solution landscape [Dor80]. The SD search typically takes several small steps while descending a narrow valley in the solution landscape thus resulting in a slow convergence rate [Pol97]. The conjugate gradient descent (CGD) search, on the other hand, is known to be one of the fastest in the class of gradient descent search-based optimization methods [HS80]. The CGD search procedure is very similar to the SD procedure except that different directions are followed while minimizing the objective function. Instead of consistently following the local downhill gradient direction (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 can proceed along any given direction vector independently of the other direction vectors. The CGD

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 $(|G| < \epsilon)$ 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;

     Compute 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 1.1: Serial CGD search algorithm

search procedure guarantees convergence to a local minimum of a quadratic function within n steps [Dor80] making it one of the fastest in the class of gradient descent search-based optimization methods [HS80]. The CGD search algorithm depicted in Figure 1.1 is based on the one presented in [PFTV88] with suitable adaptations (similar to the ones for the SD search algorithm described above) to take into account the fact that the solution space of the inter-probe spacings is constrained.

1.3.1   Computation of $\hat{\Pi}$

Determining the optimal clone ordering $\hat{\Pi}$, is an intractable problem that entails a combinatorial search through the discrete space of all possible permutations of $\{1, \ldots, n\}$. We present three stochastic discrete optimization algorithms for this purpose, Simulated Annealing (SA), Large Step Markov Chain (LSMC) and the Genetic Algorithm (GA). In particular, we chose to augment the classical GA with the stochastic hill climbing capability of the LSMC algorithm. We provide a brief description of these three algorithms.

Simulated Annealing

A single iteration of the SA algorithm consists of three basic phases: (i) perturb, (ii) evaluate, and (iii) decide. In the perturb phase, the current probe ordering Π_i is systematically perturbed to yield another candidate probe ordering Π_j by reversing the ordering within a block of probes where the endpoints of the block are chosen at random. This perturbation is referred to as a *2-opt* heuristic in the context of the TSP [LK73]. In the evaluate phase, the function $f(\Pi_j, \hat{Y}_{\Pi_j})$ (equation (1.10)) is computed. In the decide phase, the new can-

Simulated Annealing Algorithm:

1. Choose a random order of probes, Π and a sufficiently high value of the temperature T. Compute $f(\Pi, \hat{Y}_\Pi)$.

2. Choose a random block within the ordering Π.

3. Perform a block reversal and call the new ordering Π'.

4. Compute $f(\Pi', \hat{Y}_{\Pi'})$.

5. If $f(\Pi', \hat{Y}_{\Pi'}) < f(\Pi, \hat{Y}_\Pi)$, then replace the existing solution $(\Pi, \hat{Y}_\Pi)$ by the new solution $(\Pi', \hat{Y}_{\Pi'})$,
   Else if $f(\Pi', \hat{Y}_{\Pi'}) \geq f(\Pi, \hat{Y}_\Pi)$, then

   Generate a random number x that is uniformly distributed in the interval $[0, 1]$.

   If $x < \exp(-(f(\Pi', \hat{Y}_{\Pi'}) - f(\Pi, \hat{Y}_\Pi)/T)$ then, replace the existing solution $(\Pi, \hat{Y}_\Pi)$ by the new solution $(\Pi', \hat{Y}_{\Pi'})$,

   Else retain the existing solution $(\Pi, \hat{Y}_\Pi)$.

6. Repeat steps 2-5 for a given value of T until D reorderings have been attempted or S successful reorderings have been recorded, whichever comes first. An ordering is deemed successful if it lowers the objective function $f(\Pi, \hat{Y}_\Pi)$.

7. Check if the convergence criterion has been met. If yes, stop if not reduce T using the annealing schedule and go to step 2.

FIGURE 1.2: SA algorithm for computing the probe ordering and inter-probe spacings

didate solution Π_j is accepted with probability p which is computed using the Metropolis function [MRR$^+$53]:

$$p = \begin{cases} 1 & \text{if } f(\Pi_j, \hat{Y}_{\Pi_j}) < f(\Pi_i, \hat{Y}_{\Pi_i}) \\ \exp\left(-\frac{f(\Pi_j, \hat{Y}_{\Pi_j}) - f(\Pi_i, \hat{Y}_{\Pi_i})}{T}\right) & \text{if } f(\Pi_j, \hat{Y}_{\Pi_j}) \geq f(\Pi_i, \hat{Y}_{\Pi_i}) \end{cases} \tag{1.18}$$

or using the Boltzmann function $B(T)$ [AK89]:

$$p = B(T) = \frac{1}{1 + \exp\left(\frac{f(\Pi_j, \hat{Y}_{\Pi_j}) - f(\Pi_i, \hat{Y}_{\Pi_i})}{T}\right)} \tag{1.19}$$

at a given value of temperature T, whereas Π_i is retained with probability $(1 - p)$.

Both, the Metropolis function and the Boltzmann function, ensure that SA generates an asymptotically ergodic Markov chain of solution states at a given temperature value. Geman and Geman [GG84] have shown that logarithmic annealing schedules of the form $T_k = R/\log k$ for some value of $R > 0$ ensure asymptotic convergence to a global minimum with unit probability in the limit $k \to \infty$. The convergence criterion in our case was the fact that the value of the objective function had not changed for the past k successive annealing steps. However, we used a geometric annealing schedule of the form $T_{k+1} = \alpha T_k$ where α is a value less than but close to 1. We used a value of $\alpha = 0.95$ in all of our experiments. Although the aforementioned geometric annealing schedule does not strictly ensure asymptotic convergence to a global optimum as the logarithmic annealing schedule does, it is much faster and has been found to yield very good solutions in practice [RSV91]. Figure 1.2 gives the outline of the serial SA algorithm.

Large Step Markov Chain Algorithm

A single iteration of the LSMC algorithm (like the SA algorithm) also consists of the perturb, evaluate, and decide phases. The major difference between SA and the LSMC algorithms arises from the fact that the classical SA algorithm performs a strictly *local* perturbation in the perturb phase whereas the LSMC algorithm performs a non-local perturbation followed by an exhaustive local search. The LSMC algorithm combines the stochastic decision function with an exhaustive local search using the 2-opt heuristic. The current solution at every stage is guaranteed to be locally optimal under the 2-opt heuristic. In the perturb phase, the current solution (which is locally optimal) is subject to a non-local perturbation termed as a *double-bridge kick* [MOF91] which results in a transition to a non-local point in the search space. An exhaustive local 2-opt search is performed starting from this new point yielding a new local optimum. The choice between the new local optimum and the current solution is then made using the Metropolis decision function or the Boltzmann decision function as in the case of the SA algorithm.

The exhaustive local search using the 2-opt heuristic would, strictly speaking, entail the evaluation of the objective function $f(\Pi, Y)$ after each 2-opt perturbation. This would cause the LSMC algorithm to be computationally extremely intensive. As an effective compromise, the exhaustive local search is performed using a modified objective function. The column in the hybridization matrix H corresponding to a given probe could be considered as a binary hybridization signature of that probe. The modified objective function $f_D(\Pi)$ computes the sum of the Hamming distances between the binary hybridization signatures of successive probes in a given probe ordering Π. The local minimum of $f_D(\Pi)$ is sought using the 2-opt heuristic. Since the modified objective function $f_D(\Pi)$ is much easier to compute than the original objective function $f(\Pi, Y)$, the exhaustive local search is very fast. Note, that whereas the SA algorithm samples the entire search space, the LSMC algorithm samples only the space of locally optimal solutions. The Metropolis decision function or the Boltzmann decision function in the case of the LSMC algorithm is annealed in a manner similar to the SA algorithm. Figure 1.3 gives the outline of the serial LSMC algorithm. The LSMC algorithm starting from an initial solution, also generates, in the limit, an ergodic Markov chain of solution states which asymptotically converges to a stationary Boltzmann distribution [AK89]. The Boltzmann distribution asymptotically converges to a globally optimal solution when subject to the annealing process [GG84].

The Genetic Algorithm

The Genetic Algorithm (GA) [Mit96] begins with an initial ensemble or population of candidate solutions (typically represented by bit strings) 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 the genetic operators of selection, crossover and mutation. The selection operator selects two candidate solutions from the current population with probability in direct proportion to their fitness values using a *roulette-wheel* procedure. The fitness function is the negative of the NLL objective function in equation (1.6) 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. The purpose of crossover is to enable large-scale exploration of the search space [Mit96]. The child chromosomes resulting from the crossover operator are then subject to the mutation operator which represents a random local (i.e., 2-opt) perturbation in the solution space. The new generation of chromosomes, thus created us-

Large Step Markov Chain (LSMC) Algorithm:

1. Choose a random order of probes, Π and a sufficiently high value of the temperature T.

2. Perform an exhaustive local search using the 2-opt heuristic and the Hamming distance objective function on Π to yield new probe order Π'. Compute $f(\Pi', \hat{Y}_{\Pi'})$

3. Perform a non-local double-bridge perturbation on Π' to yield a new probe order Π''.

4. Perform an exhaustive local search using the 2-opt heuristic and the Hamming distance objective function on Π'' to yield new probe order Π'''. Compute $f(\Pi''', \hat{Y}_{\Pi'''})$.

5. If $f(\Pi''', \hat{Y}_{\Pi'''}) < f(\Pi', \hat{Y}_{\Pi'})$, then replace the existing solution $(\Pi', \hat{Y}_{\Pi'})$ by the new solution $(\Pi''', \hat{Y}_{\Pi'''})$,
   Else if $f(\Pi''', \hat{Y}_{\Pi'''}) \geq f(\Pi', \hat{Y}_{\Pi'})$, then

   Generate a random number x that is uniformly distributed in the interval $[0, 1]$.

   If $x < \exp(-(f(\Pi''', \hat{Y}_{\Pi'''}) - f(\Pi', \hat{Y}_{\Pi'})/T)$ then, replace the existing solution $(\Pi', \hat{Y}_{\Pi'})$ by the new solution $(\Pi''', \hat{Y}_{\Pi'''})$,

   Else retain the existing solution $(\Pi', \hat{Y}_{\Pi'})$.

6. Repeat steps 2-5 for a given value of T until D reorderings have been attempted or S successful reorderings have been recorded, whichever comes first. An ordering is deemed successful if it lowers the objective function $f(\Pi, \hat{Y}_{\Pi})$.

7. Check if the convergence criterion has been met. If yes, stop if not reduce T using the annealing schedule and go to step 2.

FIGURE 1.3: LSMC algorithm for computing the probe ordering and inter-probe spacings

ing the aforementioned genetic operators, replaces the existing population. The population replacement is repeated for several generations of the GA until the fitness value of the best chromosome in the population has not changed over the past k generations.

In our implementation of the GA, the heuristic crossover operator proposed by Jog *et al.* [JSG89] for the TSP was improvised and incorporated. The probes in the ordering are treated as nodes in an undirected graph. The edges between the nodes in the graph are weighted by the Hamming distances between the binary hybridization signatures of the corresponding probes. The heuristic crossover can be described as follows:

1. Choose a start node from one of the selected chromosomes for crossover.
2. Compare two edges leaving from the start node between two parents and choose the shorter edge; if the shorter edge leads to an illegal sequence, choose the other edge; if both edges introduce illegal sequences, choose an edge from the remaining nodes that has the shortest Hamming distance from the start node.
3. Choose the new node as the start node and repeat step 2 until a complete sequence is generated.

The heuristic crossover described above was slightly improvised in the context of our problem. If the start node is close to the end of the sequence, the crossover will not be effective in most cases if the two parental chromosomes are similar. For example, if the start node is selected at a point after which sequences of both parental chromosomes are the same, no actual exchange of parental chromosomal segments will result from the application of the heuristic crossover operator. To avoid this situation, we start from the beginning of the sequence whenever a node close to the end of the sequence is selected as the start node. This often leads to a much improved child in a single crossover operation.

In the absence of a hill climbing mechanism, the GA exhibits a slow convergence rate [Mit96].

The incorporation of *deterministic* hill climbing into the GA typically results in premature convergence to a local optimum [BZ99]. Our previous experience with the GA has shown that incorporation of a stochastic hill climbing mechanism greatly improves the asymptotic convergence of the GA to a near-globally optimal solution [BZ99]. As a consequence, the GA was enhanced with the incorporation of a *stochastic* hill climbing search similar to that of the LSMC algorithm resulting in a GA-LSMC hybrid algorithm.

In the case of the GA-LSMC hybrid algorithm, the mutation operator is implemented using the non-local double-bridge perturbation followed by an exhaustive local 2-opt search. The mutation rate is set dynamically based on the genetic variation present in the current population. During the early stages of the GA, since genetic variations in the population are relatively high, application of the heuristic crossover usually creates better offspring. Therefore the mutation rate is kept relatively low. During the later stages of the GA, when the genetic variation is depleted due to repeated selection and crossover, the mutation rate is increased to introduce more variation in the population.

Two slightly different versions of the GA-LSMC hybrid algorithm were designed and implemented. In the first version, the population replacement strategy is deterministic, i.e., the less fit parent is replaced by the child chromosome if the child chromosome has a lower NLL objective function (i.e., higher fitness function) value. In the second version, the population replacement strategy is stochastic, i.e., the the less fit parent is replaced by the child chromosome with probability p_r computed using the Boltzmann function:

$$p_r = B(T) = \frac{1}{1 + \exp\left(\frac{f(\mathbf{x}_c) - f(\mathbf{x}_p)}{T}\right)} \tag{1.20}$$

where $f(\mathbf{x}_c)$ and $f(\mathbf{x}_p)$ are the NLL objective function values associated with the child chromosome and parent chromosome, respectively. Note that both versions of the GA incorporate the hill climbing mechanism of the LSMC algorithm. Figure 1.4 depicts the GA-LSMC hybrid algorithm.

After a sufficient number of generations, the solutions in the population represent locally optimal solutions to the problem. The final solution to the optimization problem is chosen from the ensemble of locally optimal solutions. Since a larger solution space is sampled, the final solution from the GA-LSMC algorithm is potentially better than the one obtained from randomized local neighborhood search algorithms such as SA or the LSMC algorithm. However, since an ensemble of solutions has to be analyzed in each generation or iteration, the execution time per iteration is higher.

1.4   Parallel Computation of the Maximum Likelihood Estimate

Computation of the ML 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 inter-probe spacing $\hat{Y}_\Pi$ for a given probe ordering Π (equation (1.10)). 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 (1.11)). This entails parallelization of SA, LSMC or the GA for optimization in the *discrete* domain.

The GA-LSMC Hybrid Algorithm with Stochastic 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) In the case of stochastic replacement: Replace P with S^* (if mutation is performed) or S' (if mutation is not performed) with probability p_r computed using the Boltzmann function $p_r = \frac{1}{1+\exp\left(\frac{\Delta f}{T}\right)}$;

   In the case of deterministic replacement: If $\Delta f < 0$ replace P with S^* (if mutation is performed) or S' (if mutation is not performed);

   (h) Update p_m;

   (i) In the case of stochastic replacement, update the 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 1.4: Outline of the GA-LSMC hybrid algorithm with stochastic replacement

The two levels of parallel computation were implemented on a cluster of of SMPs which constitutes a hybrid platform comprising of both, shared memory and distributed memory. Each SMP is a shared-memory multiprocessor whereas the cluster of SMPs constitutes a distributed-memory computing resource. The parallel computation of the optimal inter-probe spacing $\hat{Y}_\Pi$ for a given probe ordering Π was deemed to be well suited for data parallelism using shared-memory multithreaded programming [And00]. The distributed-memory message-passing paradigm using the Message Passing Interface (MPI) software environment [Pac96] was deemed to be better suited for parallelization of SA, LSMC and the GA.

1.4.1 Parallel SA and LSMC Algorithms

Since a candidate solution in the serial SA or LSMC algorithm can be considered to be an element of an asymptotically ergodic first-order Markov chain of solution states, we formulated and implemented two models of parallel SA (pSA) and parallel LSMC (pLSMC) algorithms based on the distribution of the Markov chain of solution states on the individual processors as described below:

- The Non-Interacting Local Markov chain (NILM) pSA and pLSMC algorithms.

- The Periodically Interacting Local Markov chain (PILM) pSA and pLSMC algorithms.

In the NILM pSA and NILM pLSMC algorithms, each processor in a distributed-memory multiprocessor system runs an independent and asynchronous version of the serial SA or LSMC algorithm. In essence, there are as many Markov chains of solution states as there are physical processors within the system. At each temperature 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, the independence of the Markov chains in different processors is ensured. The evaluation and decision functions are executed concurrently on the solution state within each processor. On termination, the best solution is selected from among all the solutions available on the individual processors. The NILM model is essentially that of multiple independent searches as described in B(iii) above.

The PILM pSA and PILM pLSMC algorithms are similar to their NILM counterparts except that just before the parameter T is updated, 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. The PILM model is essentially that of multiple periodically interacting searches as described in B(iii) above.

In the case of all the four algorithms, NILM pSA, NILM pLSMC, PILM pSA and PILM pLSMC, a master process acts as the overall controlling process and runs on one of the SMPs within the MPI cluster. The master process spawns child processes on each of the other SMPs within the MPI cluster, 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, also runs its own version of the SA or LSMC algorithm just as any of its child processes. In the case of the PILM pSA/pLSMC algorithms, before the parameter T 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 LSMC 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/pLSMC algorithms receives the initial parameters from the master process and runs its local version of the SA or LSMC 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. Upon convergence each child process conveys its result to the master process.

1.4.2 Parallel Genetic Algorithm

The approach to parallelizing the GA is based on partitioning the population amongst the available processors [CP00]. 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 [CP00]. The parallel GA (pGA) was implemented using the master-slave model with MPI on a distributed-memory platform comprising of a network of SMPs. 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 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, mixes the solutions at random and redistributes the population amongst the slave processes. This periodic mixing and redistribution of the population prevents the slave processes 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.

1.4.3 Parallel Gradient Descent Search

Due to its inherent sequential nature, we deemed data parallelism to be the appropriate parallelization scheme for the CGD search algorithm. Our previous experience showed that the speedup of a data parallel implementation of the CGD search procedure on a distributed-memory multiprocessor did not scale well with an increasing number of processors [SBK01]. This was attributed to the high inter-processor communication overhead in a distributed-memory environment. Consequently, a data parallel implementation of the CGD search algorithm on a shared-memory multiprocessor using multithreaded programming was deemed more suitable.

In our current implementation, the Y and G vectors are distributed amongst the processors within a single SMP which is a shared-memory multiprocessor. Each processor performs the required operations on its local Y_{loc} and G_{loc} subvectors concurrently with the other processors. Here, $|Y_{loc}| = |Y|/N_p$ and $|G_{loc}| = |G|/N_p$ where N_p is the number of processors within a single SMP. Implementation of the parallel algorithm follows the Master/Slave model, where both the master and slaves are implemented using IEEE POSIX threads [And00]. The slave threads are responsible for most of the computation. Coordination and synchronization among the slave threads are carried out by the master thread.

Inter-thread synchronization is realized using data types mutex and semaphore from the POSIX thread (Pthread) library [And00] and the barrier function implemented by us. Mutex is used to ensure that a critical section is executed atomically. Semaphore is used to coordinate the order of execution between the master and slaves. Barrier is employed so that no thread can proceed any further until all the threads have reached the same point in their execution. This would prevent certain threads from updating the global variables when some other threads are still using them. Two types of barriers have been used in our implementation. One of the barrier types is used for coordinating the execution of the slave threads. This is useful when the computation is conducted entirely by the slave threads without any need for coordination by the master. The other barrier type is used in the computation by slave threads when the coordination and synchronization by the master thread are necessary. Each slave thread is bound to a processor so that the time spent on switching threads between processors is reduced to a minimum. The master thread is not bound to any processor since the time used by the master thread for coordination and synchronization is insignificant compared to the slave threads.

In the case of the pLSMC algorithm, the exhaustive local search using the 2-opt heuristic was also parallelized on a shared-memory SMP using multithreaded programming. However, instead of a data parallel implementation as in the case of the CGD search procedure, a control parallel scheme for the exhaustive local search was designed. The control parallel

scheme entails multiple independent searches performed by multiple threads each bound to a distinct processor. For a given ordering of n clones, an exhaustive local search using the 2-opt heuristic would result in generation and evaluation of $O(n^2)$ distinct perturbations [LK73]. In the control parallel scheme, the generation and evaluation of distinct perturbations in carried out concurrently by multiple threads, each thread bound to a distinct processor within an SMP. The control parallel scheme follows the master-slave model wherein the master thread spawns multiple slave threads, binds each slave thread to a distinct processor within the SMP, assigns each slave thread a distinct partition of the search space (i.e., space of $O(n^2)$ distinct perturbations), collects the final result from each of the slave threads and designates the best result as the locally optimal solution under the 2-opt heuristic.

1.4.4 A Two-tier Approach for Parallel Computation of the ML Estimator

In order to ensure a modular, flexible and 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 Π using the parallel multithreaded CGD search algorithm. The coarser or upper level of parallelization pertains to the computation of $\hat{\Pi}$ using SA, LSMC or the GA.

At the coarser level, the user has a choice of using either the parallel SA (pSA), parallel LSMC (pLSMC) or the parallel GA (pGA). The parallelization of the CGD search algorithm at the finer level is transparent to pSA, pLSMC or the pGA at the coarser level, i.e., the communication and control scheme for the parallel evolutionary methods is independent of that of the parallel multithreaded CGD search algorithm. For example, one could use the serial or parallel version of the CGD search algorithm (or, for that matter, any other serial or parallel algorithm for continuous optimization at the finer level) without having to make any changes to pSA, pLSMC or the pGA at the coarser level and vice versa.

A parallel multithreaded CGD search process is embedded within each of the pSA, pLSMC and pGA processes. When the parallel multithreaded CGD search procedure is invoked from within the master or slave pSA, pLSMC or pGA process, a new set of CGD threads is spawned on the available processors within an SMP. The master CGD thread runs on the same processor as the parent pSA, pLSMC or pGA process (master or child). The master and slave CGD threads cooperate to evaluate and optimize the objective function $f_\Pi(Y)$. Having optimized $f_\Pi(Y)$, the master and slave CGD threads are terminated by the parent pGA process and the corresponding processors are available for future computation.

The two-tier approach to parallelization of the ML estimator can be seen to induce a logical tree-shaped interconnection network on the available processors within the SMP cluster. The first level is the collection of SMPs that comprise the cluster. These SMPs run the (master and slave) pSA, pLSMC or pGA processes. The second level is the collection of processors within each SMP that run the master and slave CGD threads spawned by the pSA, pLSMC or pGA process running on that SMP. The processors that run the CGD threads are connected to the processor running the parent pSA, pLSMC or pGA process but are independent of the processors running other pSA, pLSMC or pGA processes.

1.5 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-

TABLE 1.1   Specifications of the simulated clone-probe hybridization data

Data Set	n	k	N	M	ρ	η
1	50	300	180	3	2%	2%
2	100	650	850	7	2%	2%
3	200	1300	1480	7	2%	2%
4	500	3250	3700	7	2%	2%

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 [SBK01] as well as real data from *cosmid2* ($n = 109$, $k = 2046$) and *cosmid3* ($n = 111$, $k = 1937$) of the fungal genome *Neurospora crassa* made available by the Fungal Genome Resource, Department of Genetics, The University of Georgia. The simulated data were used with the specifications outlined in Table 1.1. The simulated data were generated using a program described in [SBK01] which generates clonal data of a given length with the left endpoints of the clones and probes uniformly distributed along the length of a simulated chromosome.

The pGA was implemented with the following parameters: the population size N_{pop} was chosen to be 40 for all the tests, the initial temperature T_{init} was chosen to be 1 for the pGA with stochastic replacement, the annealing factor α 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 generations (i.e., the population remained the same for two successive generations). Note that this 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}, the 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 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 pSA and pLSMC algorithms were implemented with following parameters: the initial value for the temperature was chosen to be 1.0, 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 annealing factor α in the geometric annealing function was chosen to be 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. When comparing the various parallel versions of the SA and LSMC algorithms, the product of the number of SMPs i.e., N_{SMP} and the maximum number of iterations D performed by a processor in a single annealing step was kept constant i.e., $D = (100 \cdot n)/N_{SMP}$. This ensured that the overall workload remained constant as the number of SMPs was varied thus enabling one to examine the scalability of the speedup and efficiency of the algorithms for a given problem size with increasing number of processors. In the NILM pSA and
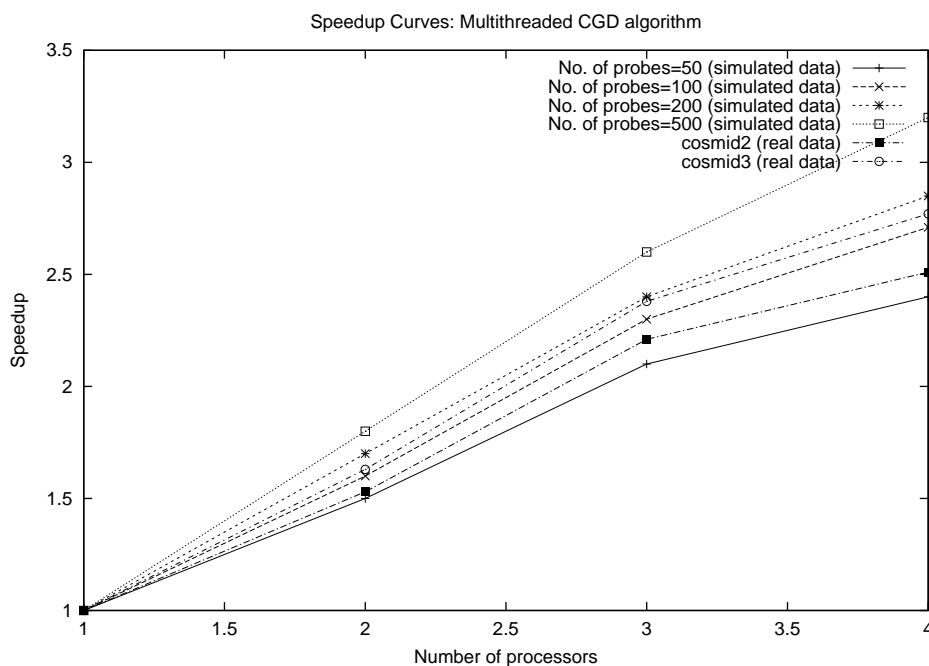
FIGURE 1.5: Speedup curves for the MTCGD algorithm on an SMP.

NILM pLSMC 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 pLSMC algorithms, each process was terminated when the number of successful perturbations in an annealing step equaled 0 for *all* the processes i.e., the master process and all the child processes. This condition was checked during the synchronization phase at the end of each annealing step.

The parallel multithreaded CGD (MTCGD) search algorithm was tested on the simulated data set in Table 1.1 and on real data derived from *cosmid2* ($n = 109, k = 2046$) of the fungal genome *Neurospora crassa*. Figure 1.5 shows the speedup of the MTCGD search algorithm for varying N_p (i.e., number of processors within an SMP). As can be seen, the payoff in the parallelization of the CGD search algorithm is better realized for larger values of n (i.e., larger problem sizes). For $n = 200$ and $n = 500$, the best efficiency figures were seen to be in the range 93%–95% whereas for $n = 50$ the best efficiency figure were about 85%. For a given problem size, the efficiency was seen to decrease with an increasing number of processors N_p within the SMP. This is expected since the overhead imposed by barrier synchronization, mutex locks and semaphores increasingly dominates the overall execution time as N_p is increased for a given problem size.

The parallel multithreaded exhaustive local search algorithm was also tested on the simulated data set in Table 1.1. Figure 1.6 shows the speedup of the multithreaded exhaustive local search algorithm for varying N_p (i.e., number of processors within an SMP). For a given problem size, the efficiency was 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 no frequent synchronization amongst the slave threads or between the master thread and the slave threads. Since each slave thread
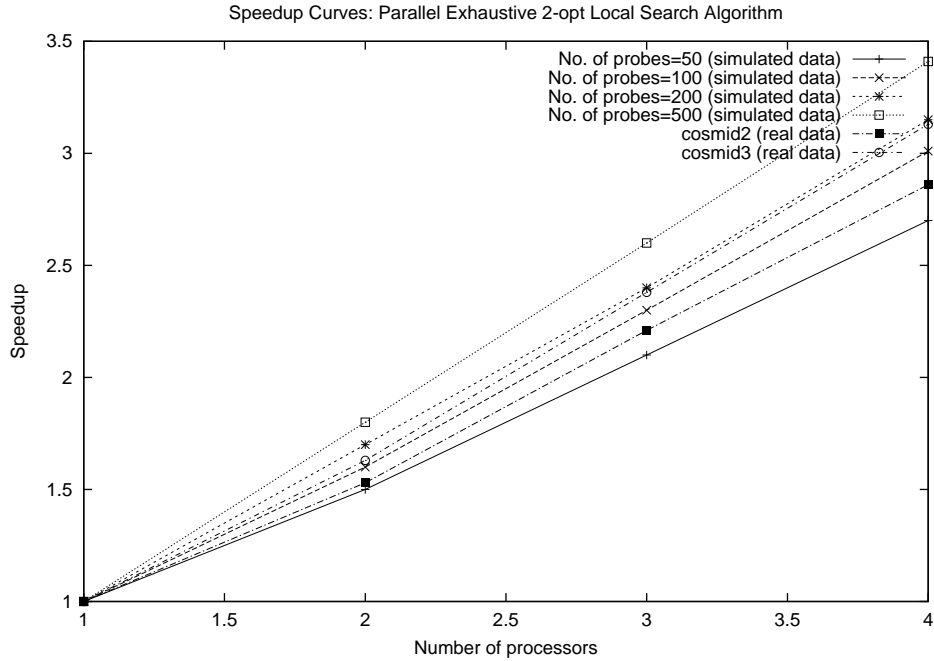
FIGURE 1.6: Speedup curves for the multithreaded exhaustive local search algorithm on an SMP.

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 more frequent.

Figure 1.7 shows the speedup of the PILM pSA algorithm on the simulated and real data sets. Likewise, Figure 1.8 shows the speedup of the PILM pLSMC algorithm on the simulated and real data sets. The PILM versions of the pSA and pLSMC algorithms were observed to exhibit superior performance when compared to their NILM counterparts. This was expected since the PILM pSA and PILM pLSMC algorithms focus on the more promising regions of the search space via periodic synchronization. As can be observed, the pSA and pLSMC algorithms exhibit consistent and scalable speedup with an increasing total number of processors $N_{proc} = N_p \cdot N_{SMP}$. As expected, the speedup scales better with increasing number of processors for larger values of the number of probes n (i.e., larger problem sizes). Overall, the pSA algorithm was seen to scale better than the PLSMC algorithm. The reason for this is that the serial LSMC algorithm is much faster than the serial SA algorithm [SBK01]. This implies that the pLSMC algorithm is better suited for larger problem instances than the pSA algorithm.

Figure 1.9 shows the speedup of the parallel GA-LSMC (pGA-LSMC) algorithm with stochastic replacement. The performance of the pGA-LSMC algorithm with stochastic replacement was observed to be superior to the pGA-LSMC algorithm with deterministic replacement in that the former was observed to yield better solutions with a smaller convergence time in most cases. This is in conformity with the results from our earlier work [BZ99] which showed that the incorporation of a stochastic hill climbing mechanism (which, in the case of the pGA, is manifested in the form of stochastic replacement using the Boltzmann
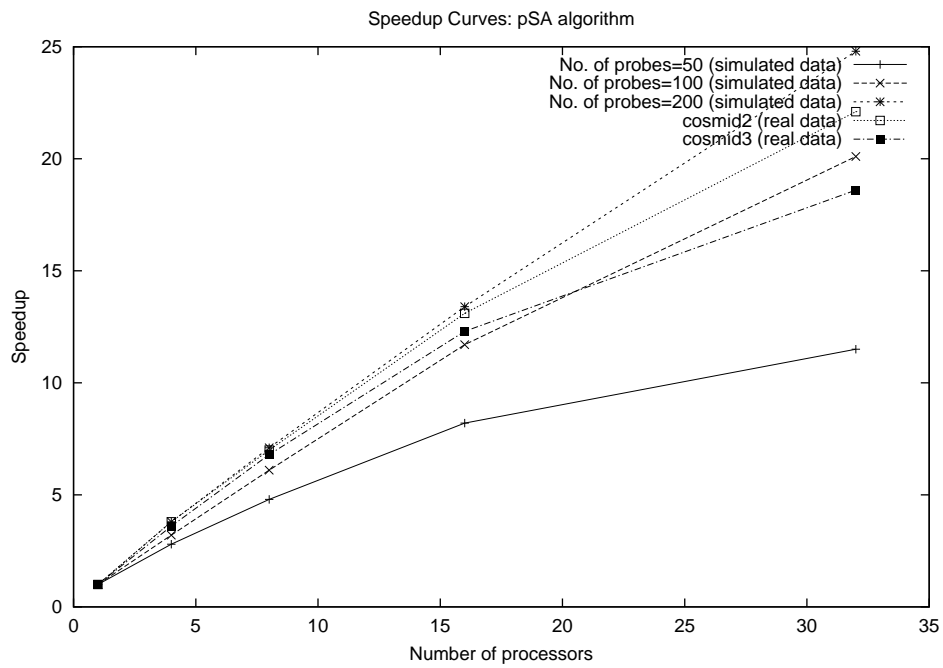
FIGURE 1.7: Speedup curves for the PILM pSA algorithm on an SMP.
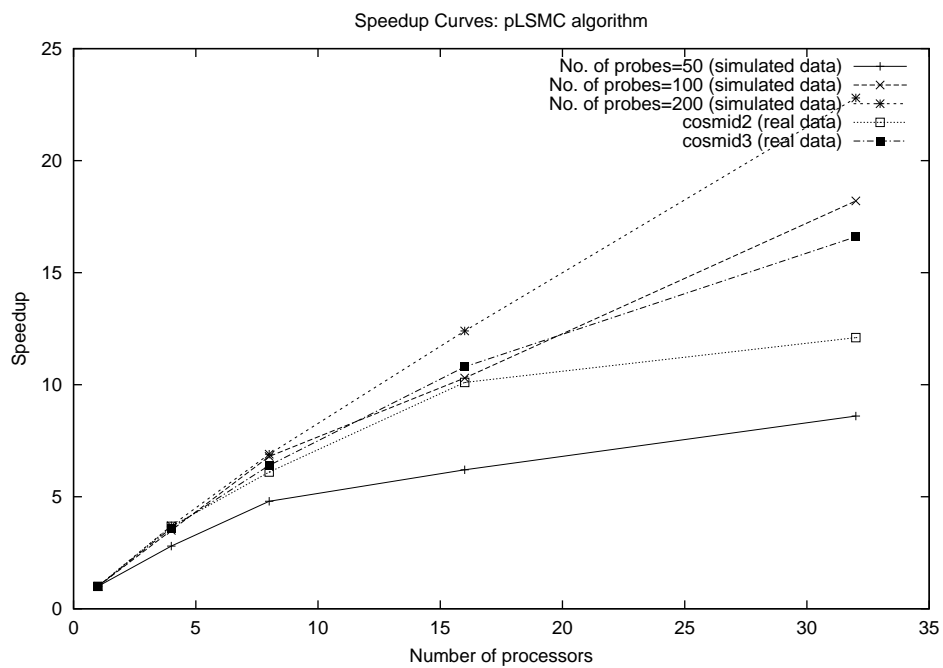

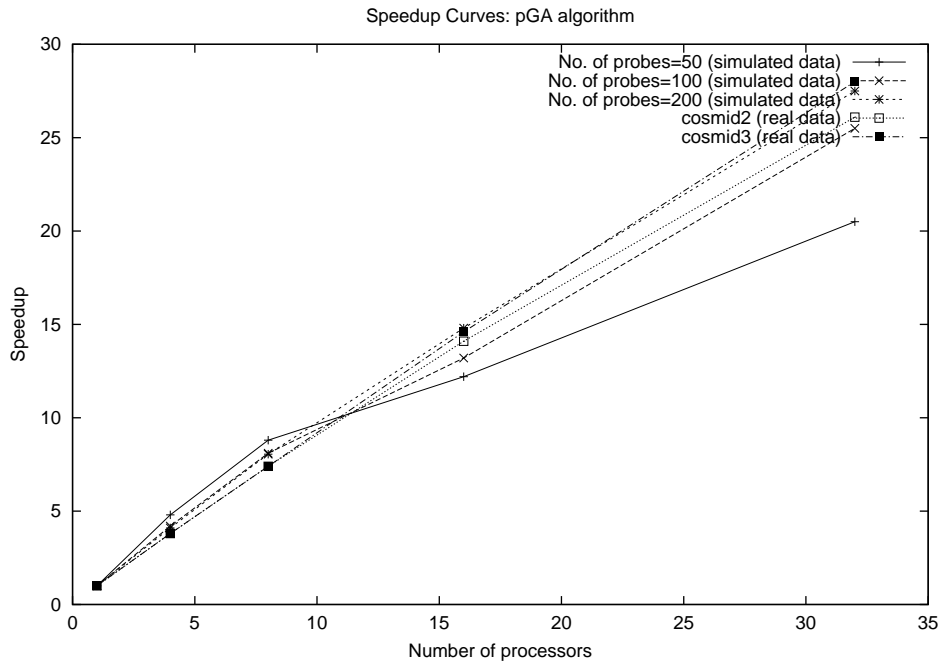FIGURE 1.8: Speedup curves for the PILM pLSMC algorithm on an SMP.

FIGURE 1.9: Speedup curves for the pGA-LSMC algorithm on an SMP.

function), does improve the convergence rate and the ability of the pGA to explore more promising regions of the search space (thus resulting in a better final solution). As in the case of the pSA and pLSMC algorithms, the speedup of the pGA-LSMC algorithm was seen to scale better with increasing number of processors for larger values of n (i.e., larger problem sizes).

The pGA-LSMC algorithm was also observed to exhibit superlinear speedup in some instances. This can be attributed to two causes, population caching and the stochastic nature of the search process in the serial and parallel GA. Population caching is 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. When the number of processors is small, the caching effect can overcome the inter-processor communication and synchronization overhead resulting in superlinear 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 (both, stochastic and deterministic) such as SA, LSMC and branch-and-bound [LS84].

The efficiency values for all versions of all the three parallel algorithms (PSA, pLSMC and pGA-LSMC) are observed to be higher for larger problem instances (larger values of n and k) for a given value of N_{proc}. The efficiency values also show an overall declining trend for increasing values of N_{proc} for a given problem instance for all versions of all the three parallel algorithms. These observations are in conformity with the general expectations regarding the performance of parallel stochastic optimization algorithms.

The results of the serial GA-LSMC hybrid algorithm with stochastic replacement were compared with those of the serial SA and serial LSMC algorithms (Table 1.2). The serial

TABLE 1.2   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
cosmid2	27962	12731.37	-	34704	12757.55	-	108499	12949.99	-
cosmid3	14922	12584.62	-	30183	12501.88	-	45533	13212.85	-

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

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 1.2 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 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 a physical map with fewer and longer contigs (i.e., fewer contig breaks) than the SA and LSMC algorithms suggesting that the GA-LSMC algorithm is capable of yielding solutions of higher quality. This clearly showed that the heuristic crossover in conjunction with population sampling is a powerful mechanism for exploration of the search space in the case of the GA. Note that whereas the SA and LSMC algorithms possess stochastic hill climbing capabilities, they do not possess the capability for large-scale exploration of the search space via population sampling and the crossover operator as does the GA.

1.6   Conclusions and Future Directions

In this chapter we presented an information theoretic approach to genome reconstruction. Information theoretic reconstruction approaches based on the maximum likelihood (ML) model or the Bayesian maximum a posteriori (MAP) model have been used extensively in image and signal reconstruction. In this chapter, we described 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 estimator optimizes a likelihood function defined over the spacings and orderings of probes under an experimental protocol wherein clones of equal length are hybridized to a maximal subset of non-overlapping equal-length clones termed as probes. The estimation procedure was shown to involve a combination of continuous and discrete optimization; the former to determine a set of optimal inter-probe spacings for a given probe ordering and the latter to determine the optimal probe ordering. The conjugate gradient descent (CGD) search 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 Simulated Annealing (SA), Large Step Markov Chain (LSMC) and the Genetic Algorithm (GA). In particular, the incorporation of stochastic hill climbing into the traditional GA was shown to result in a GA-LSMC hybrid algorithm with convergence behavior superior to the traditional GA.

The problem of ML estimation-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 CGD search procedure was parallelized at the lower level and the SA, LSMC or the GA was simultaneously parallelized at the higher level. The parallel algorithms were implemented on a networked cluster of shared-memory symmetric multiprocessors (SMPs) running the Message Passing Interface (MPI) environment. The cluster of SMPs offered a hybrid of shared-memory (within a single SMP) and distributed-memory (across the SMP cluster) parallel computing. A shared-memory data parallel approach where the components of the gradient vector are distributed amongst the individual processors within an SMP was deemed more suitable for the parallelization of the CGD search procedure. A distributed-memory control parallel scheme where individual SMPs perform noninteracting or periodically interacting searches was deemed more suitable for the parallelization of SA, LSMC and the GA.

Our experimental results on simulated clone-probe data showed that the payoff in data parallelization of the CGD procedure was better realized for large problem sizes (i.e., large values of n and k). A similar trend was observed in the case of the parallel versions of SA, LSMC and the GA. In the case of the parallel GA, superlinear speedup was observed in some instances which could be attributed to population caching effects. The parallel exhaustive local 2-opt search algorithm was observed to be the the most scalable in terms of speedup. This was expected since the parallel exhaustive local 2-opt search algorithm entails minimal interprocessor communication and synchronization overhead. Overall, the experimental results were found to be in conformity with expectations based on formal analysis.

Future research will investigate extensions of the ML function that also encapsulate errors due to repeat DNA sequences in addition to false positives and false negatives. The current MPI implementation of the ML estimator is targeted towards a homogeneous distributed processing 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 distributed processing platform such as a network of SMPs that differ in processing speeds, a scenario that is more likely to be encountered in the real world. Other combinatorial optimization techniques such as those based on the Lagrangian-based global search and tabu search will also be investigated. Extensions of the ML estimation-based approach to a Bayesian a-posteriori estimation-based approach, where the ordering of clones/probes containing genetic markers as inferred from a genetic map are used as a prior distribution, will also be investigated.

Acknowledgments

References

References

[AK89] E.H.L. Aarts and K. Korst. *Simulated Annealing and Boltzman Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing.* Wiley, New York, NY, 1989.

[And00] G. Andrews. *Foundations of Multithreaded, Parallel, and Distributed Programming*. Addison Wesley Pub. Co., Reading, MA, 2000.

[Arn97] J. Arnold. Editorial. *Fungal Genetics and Biology*, 21:254–257, 1997.

[Bal94] D.J. Balding. Design and analysis of chromosome physical mapping experiments. *Philos. Trans. Roy. Soc. London Ser. B.*, 334:329–335, 1994.

[BDC97] A. Ben-Dor and B. Chor. On constructing radiation hybrid maps. In *Proc. ACM Conf. Computational Molecular Biology*, pages 17–26, 1997.

[BZ99] S.M. Bhandarkar and H. Zhang. Image segmentation using evolutionary computation. *IEEE Trans. Evolutionary Computation*, 3(1):1–21, April 1999.

[CAT92] A. J. Cuticchia, J. Arnold, and W. E. Timberlake. The use of simulated annealing in chromosome reconstruction experiments based on binary scoring. *Genetics*, 132:591–601, 1992.

[CJK⁺97] T. Christof, M. Jünger, J.D. Kececioglu, P. Mutzel, and G. Reinelt. A branch–and–cut approach to physical mapping of chromosomes by unique end probes. *Jour. Computational Biology*, 4(4):433–447, 1997.

[CK99] T. Christof and J.D. Kececioglu. Computing physical maps of chromosomes with non-overlapping probes by branch–and–cut. In *Proc. ACM Conf. Computational Molecular Biology*, pages 115–123, Lyon, France, April 1999.

[CP00] E. Cantu-Paz. *Efficient and Accurate Parallel Genetic Algorithms*. Kluwer Academic Publishers, Boston, MA, November 2000.

[CS95] B. Chor and M. Sudan. A geometric approach to betweenness. In *Proc. European Symp. Algorithms*, volume 979, pages 227–237. Springer–Verlag Lecture Notes in Computer Science, 1995.

[Dor80] C.N. Dorny. *A Vector Space Approach to Models and Optimization*. R.E. Krieger Publishing Company, Huntington, NY, 1980.

[DWP97] D.S. Greenberg D.B. Wilson and C.A. Phillips. Beyond islands: runs in clone–probe matrices. In *Proc. ACM Conf. Computational Molecular Biology*, pages 320–329, 1997.

[FAW95] L.A. Newberg F. Alizadeh, R.M. Karp and D.K. Weisser. Physical mapping of chromosomes: a combinatorial problem in molecular biology. *Algorithmica*, 13(1/2):52–76, 1995.

[FAZ94] D.K. Weisser F. Alizadeh, R.M. Karp and G. Zweig. Physical mapping of chromosomes using unique probes. In *Proc. ACM–SIAM Conf. Discrete Algorithms*, pages 489–500, 1994.

[FJK⁺97] D.P. Fasulo, T. Jiang, R.M. Karp, R. Settergren, and E.C. Thayer. An algorithmic approach to multiple complete digest mapping. In *Proc. ACM Conf. Computational Molecular Biology*, pages 118–127, 1997.

[FTA92] Y.X. Fu, W.E. Timberlake, and J. Arnold. On the design of genome mapping experiments using short synthetic oligonucleotides. *Biometrics*, 48:337–359, 1992.

[GG84] S. Geman and D. Geman. stochastic relaxation, gibbs distribution and the bayesian restoration of images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

[GI95] D.S. Greenberg and S. Istrail. Physical mapping by sts hybridization: algorithmic strategies and the challenge of software evaluation. *Jour. Computational Biology*, 2(2):219–273, 1995.

[GJ79] M.S. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP–Completeness*. W.H. Freeman, New York, NY, 1979.

[Had23] H. Hadamard. *Lectures on the Cauchy Problem in Linear Partial Differential Equations*. Yale University Press, New Haven, CT, 1923.

[HS80]   M. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Jour. Research of National Bureau of Standards*, 49:409–436, 1980.

[JK97]   T. Jiang and R.M. Karp. Mapping clones with a given ordering or interleaving. In *Proc. ACM–SIAM Conf. Discrete Algorithms*, pages 400–409, 1997.

[JM97]   M. Jain and E.W. Myers. Algorithms for computing and integrating physical maps using unique probes. *Jour. Computational Biology*, 4(4):449–466, 1997.

[JSG89] 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. In *Proc. Intl. Conf. Genetic Algorithms*, pages 110–115, Fairfax, VA, June 1989.

[KM95]   J.D. Kececioglu and E.W. Myers. Combinatorial algorithms for dna sequence assembly. *Algorithmica*, 13:7–51, 1995.

[KS98]   R.M. Karp and R. Shamir. Algorithms for optical mapping. In *Proc. ACM Conf. Computational Molecular Biology*, pages 117–124, 1998.

[KSA00] J.D. Kececioglu, S.S. Shete, and J. Arnold. Reconstructing distances in physical maps of chromosomes with nonoverlapping probes. In *Proc. ACM Conf. Computational Molecular Biology*, pages 183–192, Tokyo, Japan, April 2000.

[LDW98] J.K. Lee, V. Dancik, and M.S. Waterman. Estimation for restriction sites observed by optical mapping using reversible-jump markov chain monte carlo. In *Proc. ACM Conf. Computational Molecular Biology*, pages 147–152, 1998.

[LG87]   E.S. Lander and P. Green. Construction of multi-locus genetic linkage maps in humans. *Proc. Natl. Acad. Sci.*, 84:2363–2367, 1987.

[LK73]   S. Lin and B. Kernighan. An effective heuristic search algorithm for the traveling salesman problem. *Operations Research*, 21:498–516, 1973.

[LS84]   T.H. Lai and S. Sahni. Anomalies in parallel branch and bound algorithms. *Comm. ACM*, 27(6):594–602, June 1984.

[LW88]   E.S. Lander and M.S. Waterman. Genomic mapping by fingerprinting random clones: a mathematical analysis. *Genomics*, 2:231–239, 1988.

[MGM+93] R. Mott, A.V. Grigoriev, E. Maier, J.D. Hoheisel, and H. Lehrach. Algorithms and software tools for ordering clone libraries: application to the mapping of the genome *s. pombe*. *Nucleic Acids Research*, 21(8):1965–1974, 1993.

[Mit96] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, 1996.

[MOF91] O. Martin, S.W. Otto, and E.W. Felten. Large-step markov chains for the traveling salesman problem. *Complex Systems*, 5(3):299–326, 1991.

[MP97]   S. Muthukrishnan and L. Parida. Towards constructing physical maps by optical mapping: an effective, simple, combinatorial approach. In *Proc. ACM Conf. Computational Molecular Biology*, pages 209–219, 1997.

[MRR+53] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Jour. Chemical Physics*, 21:1087–1092, 1953.

[NS94]   D.O. Nelson and T.P. Speed. Statistical issues in constructing high resolution physical maps. *Statistical Science*, pages 334–354, 1994.

[Pac96] P. Pacheco. *Parallel Programming with MPI*. Morgan Kaufmann Publishers, San Francisco, CA, 1996.

[PFTV88] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, New York, NY, 1988.

[Pol97] E. Polak. Optimization: Algorithms and consistent approximations. *Applied Mathematical Sciences*, 124, 1997.

[RAW91] S. Tavare R. Arratia, E.S. Lander and M.S. Waterman. Genomic mapping by

anchoring random probes: a mathematical analysis. *Genomics*, 11:806–827, 1991.

[RSV91]   F. Romeo and A. Sangiovanni-Vincentelli. A theoretical framework for simulated annealing. *Algorithmica*, 6:302–345, 1991.

[SBK01]   S.S. Shete S.M. Bhandarkar, S.A. Machaka and R.N. Kota. Parallel computation of a maximum likelihood estimator of a physical map. *Genetics, special issue on Computational Biology*, 157(3):1021–1043, March 2001.

[SKSL97] D. Slonim, L. Kruglyak, L. Stein, and E. Lander. Building human genome maps with radiation hybrids. In *Proc. ACM Conf. Computational Molecular Biology*, pages 277–286, 1997.

[Stu13]   A.H. Sturtevant. The linear arrangement of six sex-linked factors in drosophila as shown by their mode of association. *Jour. Exp. Zool.*, 14:43–49, 1913.

[WPG⁺94] Y. Wang, R.A. Prade, J. Griffith, W.E. Timberlake, and J. Arnold. A fast random cost algorithm for physical mapping. *Proc. Natl. Acad. Sci.*, 91:11094–11098, 1994.

[XCP⁺96] M. Xiong, H.J. Chen, R.A. Prade, Y. Wang, J. Griffith, W.E. Timberlake, and J. Arnold. On the consistency of a physical mapping method to reconstruct a chromosome *in vitro*. *Genetics*, 142(1):267–284, 1996.

[ZM93]   M.Q. Zhang and T.G. Marr. Genome mapping by nonrandom anchoring: a discrete theoretical analysis. In *Proc. ACM Conf. Computational Molecular Biology*, volume 90, pages 600–604, 1993.

Index

2-opt heuristic, 1-10

Boltzmann function, 1-9

conjugate gradient descent search, 1-7
contig, 1-22

exhaustive local search, 1-10

genetic algorithm, 1-8
genetic map, 1-1
genetic operators, 1-10

heuristic crossover operator, 1-11
hill climbing, 1-8
hybridization, 1-2
hybridization matrix, 1-2

large step Markov chain, 1-8

maximum likelihood estimator, 1-4
Metropolis function, 1-9
MPI, 1-13
multithreaded programming, 1-3, 1-15

negative log-likelihood function, 1-5

physical map, 1-1
physical mapping protocol, 1-2
POSIX, 1-15

shared-memory multiprocessor, 1-13
simulated annealing, 1-8
SMP, 1-3, 1-13
steepest descent search, 1-6