The Dissertation Committee for Xiaohu Shen
certifies that this is the approved version of the following dissertation:

Bayesian Inference Methods for Next Generation DNA Sequencing

Committee:

Haris Vikalo, Supervisor

Gustavo de Veciana

Sriram Vishwanath

Sujay Sanghavi

Pradeep Ravikumar

Bayesian Inference Methods for Next Generation DNA Sequencing

by

Xiaohu Shen, B.E., M.S.E.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2014

Dedicated to my family.

Acknowledgments

During my last several years as a graduate student at the University of Texas at Austin, I am very lucky to work and live with a number of great people. Without the support and help from them, this dissertation would not be possible.

I am extremely thankful to my advisor, Prof. Haris Vikalo, who has influenced and helped me throughout every aspect of my research. Dr. Vikalo is a great advisor and mentor. He taught me how to do research from the beginning. His deep insight and experience in biomedical signal processing and algorithms inspired the ideas of the research presented in this dissertation. None of these work can be completed without his advice on the methodologies and enormous revisions to my writing. His great patience and vision always encourage me to pursuit the best in research and future career.

I would like to thank my committee members: Prof. Gustavo de Veciana, Prof. Sriram Vishwanath, Prof. Sujay Sanghavi and Prof. Pradeep Ravikumar, for their time and valuable advice. A lot of the methodologies used in my research are inspired by the courses they offered at UT Austin.

In my PhD life, I greatly enjoyed the collaborations with my labmates: Manohar Shamaiah, Shreepriya Das, Somsubhra Barik, Soyeon Ahn and Natalia Arzeno-Gonzales. I am also very glad to have best friends around. Thanks

to all of you: Aibo Tian, Fei Zhou, Chun Wang, Ping Xia, Yuhuan Du, Zheng Lu, Ye Chen, Hailong Xiao, Hongbo Si, Lei Guo and Kai Yang.

Lastly but most importantly, I thank my parents and my sister, for their constant support and encouragement throughout my life.

Bayesian Inference Methods for Next Generation DNA Sequencing

Publication No. _____

Xiaohu Shen, Ph.D.
The University of Texas at Austin, 2014

Supervisor: Haris Vikalo

Recently developed next-generation sequencing systems are capable of rapid and cost-effective DNA sequencing, thus enabling routine sequencing tasks and taking us one step closer to personalized medicine. To provide a blueprint of a target genome, next-generation sequencing systems typically employ the so called shotgun sequencing strategy and oversample the genome with a library of relatively short overlapping reads. The order of nucleotides in the short reads is determined by processing acquired noisy signals generated by the sequencing platforms, and the overlaps between the reads are exploited to assemble the target long genome. Next-generation sequencing utilizes massively parallel array-based technology to speed up the sequencing and reduce the cost. However, accuracy and lengths of the short reads are yet to surpass those provided by the conventional slower and costlier Sanger sequencing method.

In this thesis, we first focus on Illumina's sequencing-by-synthesis platform which relies on reversible terminator chemistry and describe the acquired signal by a Hidden Markov Model. Relying on this model and sequential Monte Carlo methods, we develop a parameter estimation and base calling scheme called ParticleCall. ParticleCall is tested on an experimental data set obtained by sequencing phiX174 bacteriophage using Illumina's Genome Analyzer II. The results show that ParticleCall scheme is significantly more computationally efficient than the best performing unsupervised base calling method currently available, while achieving the same accuracy.

Having addressed the problem of base calling of short reads, we turn our attention to genome assembly. Assembly of a genome from acquired short reads is a computationally daunting task even in the scenario where a reference genome exists. Errors and gaps in the reference, and perfect repeat regions in the target, further render the assembly challenging and cause inaccuracies. We formulate reference-guided assembly as the inference problem on a bipartite graph and solve it using a message-passing algorithm. The proposed algorithm can be interpreted as the classical belief propagation scheme under a certain prior. Unlike existing state-of-the-art methods, the proposed algorithm combines the information provided by the reads without needing to know reliability of the short reads (so-called quality scores). Relation of the message-passing algorithm to a provably convergent power iteration scheme is discussed. Results on both simulated and experimental data demonstrate that the proposed message-passing algorithm outperforms commonly used state-of-the-art tools,

and it nearly achieves the performance of a genie-aided maximum a posteriori (MAP) scheme.

We then consider the reference-free genome assembly problem, i.e., the *de novo* assembly. Various methods for *de novo* assembly have been proposed in literature, all of whom are very sensitive to errors in short reads. We develop a novel error-correction method that enables performance improvements of *de novo* assembly. The new method relies on a suffix array structure built on the short reads data. It incorporates a hypothesis testing procedure utilizing the sum of quality information as the test statistic to improve the accuracy of overlap detection.

Finally, we consider an inference problem in gene regulatory networks. Gene regulatory networks are highly complex dynamical systems comprising biomolecular components which interact with each other and through those interactions determine gene expression levels, i.e., determine the rate of gene transcription. In this thesis, a particle filter with Markov Chain Monte Carlo move step is employed for the estimation of reaction rate constants in gene regulatory networks modeled by chemical Langevin equations. Simulation studies demonstrate that the proposed technique outperforms previously considered methods while being computationally more efficient. Dynamic behavior of gene regulatory networks averaged over a large number of cells can be modeled by ordinary differential equations. For this scenario, we compute an approximation to the Cramer-Rao lower bound on the mean-square error of estimating reaction rates and demonstrate that, when the number of unknown parameters

is small, the proposed particle filter can be nearly optimal.

In summary, this thesis presents a set of Bayesian inference methods for base-calling and sequence assembly in next-generation DNA sequencing. Experimental studies shows the advantage of proposed algorithms over traditional methods.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The advancements of next-generation sequencing technologies have enabled inexpensive and rapid generation of vast amounts of sequencing data [1–3]. At the same time, high-throughput sequencing technologies present us with the challenge of processing and analyzing large data sets that they provide. Development of these analysis methods can improve the efficiency and accuracy of next-generation sequencing, enable routine sequencing tasks and take us one step closer to personalized medicine.

A fundamental computational challenge encountered in next-generation sequencing systems is the one of determining the order of nucleotides from the acquired measurements, the task typically referred to as *base calling*. The accuracy of base calling is of essential importance for various downstream applications including sequence assembly, SNP calling, and genotype calling [4]. Moreover, improving base calling accuracy may enable achieving desired performance of downstream applications with smaller sequencing coverage, which translates to a reduction in the sequencing cost.

In re-sequencing tasks, short reads data obtained from base-calling is used to assemble the target genome with the goal of, e.g., studying genet-

ic variations. Assembly of a genome from short reads is a computationally challenging task even in the scenario where a reference genome exists. Errors and gaps in the reference, and perfect repeat regions in the target, render target assembly difficult. When reconstructing the target DNA sequence using short-reads and a reference, the short reads are first mapped to a reference sequence (a DNA sequence highly similar to the target but not identical) using an alignment algorithm (e.g., [5], [6]). Then each position along the target is determined by combining information provided by all the reads that cover that particular position. Due to the errors in base calls, short length of the reads, and repetitiveness in the target, both the mapping and the sequence assembly steps are potentially erroneous. Such errors can be combated by using redundancy – each base in the target sequence is typically covered by a large number of reads.

When a reference genome is not available, the target genome needs to be reconstructed directly from the short reads. This problem is called *de novo* assembly. Several *de novo* assembly algorithms have been proposed recently [7–10]. These algorithms are highly sensitive to read errors. To improve the accuracy of the assembly using next-generation sequencing data, a key step is to correct the errors in the raw short reads using the redundancy information provided by the high coverage of next-generation sequencing.

The development in DNA and RNA sequencing technology sparked a surge of interest in studying gene regulatory mechanisms. The experimental advances have been accompanied by the theoretical developments in modeling

and computational studies of the networks. Combination of these research efforts provides critical information about the functionality of cells and organisms, reveals mechanisms of genetic diseases, enables optimization of diagnostic techniques and therapies, and provides aid in the process of drug discovery.

1.1 Organization of the thesis

In Chapter 2, we develop a Hidden Markov Model (HMM) representation of the signal acquired by Illumina's sequencing-by-synthesis platforms and develop a particle filtering (i.e., sequential Monte Carlo) base-calling scheme that we refer to as ParticleCall [11, 12]. When relying on the BayesCall's Markov Chain Monte Carlo implementation of the EM algorithm (MCEM) to estimate system parameters, ParticleCall achieves the same error rate performance as BayesCall while reducing the time needed for base calling by a factor of 3. To improve the speed of parameter estimation, we develop a particle filter implementation of the EM algorithm (PFEM). PFEM significantly reduces parameter estimation time while leading to a very minor deterioration of the accuracy of base calling. We demonstrate that ParticleCall also has the best discrimination ability among all of the considered base calling schemes.

In Chapter 3, we formulate the reference-guided assembly problem as the inference of the genome sequence on a bipartite graph and solve it using a message-passing algorithm [13–15]. Unlike existing state-of-the-art methods, the proposed algorithm performs reference-guided sequence assembly without relying on the, possibly inaccurate, quality scores of the short reads. Instead,

it infers reliability of a base in the assembled sequence by combining the information of all the reads covering that particular position. The proposed algorithm can be interpreted as the classical belief propagation under a certain prior. Binary reformulation of the problem leads to an alternative solution in the form of another message passing algorithm that is closely related to the power iteration method. The power iteration method approximates the solution to the sequence assembly problem by the leading singular vector of a matrix comprising read data. The power iteration method has guaranteed convergence, and its careful examination provides relation between the algorithm accuracy and the number of iterations. To evaluate the performance of the proposed techniques, we compare them with a genie-aided maximum a posteriori (MAP) sequence assembly scheme which is an idealized assembler with perfect quality score information and error-free mapping of the reads to their locations. Results on both simulated and experimental data obtained by sequencing *Escherichia Coli* and *Neisseria Meningitidis* at UT Austin's Center for Genomic Sequencing and Analysis demonstrate that our proposed message-passing algorithm performs close to the aforementioned genie-aided MAP assembly scheme and is superior compared to state-of-the-art methods (in particular, it outperforms the aforementioned SAMtools and GATK software packages).

In Chapter 4, we consider the *de novo* sequence assembly case in which a reference genome is not available. We develop a short-reads error correction algorithm using realistic read error profiles. The proposed error correction al-

gorithm utilizing the base quality information provided by the next-generation DNA sequencing platform and uses suffix array data structure to efficiently identify the repetitive regions between the short reads. The erroneous bases are then corrected by comparing them with the bases in other reads. In this proposed algorithm, a hypothesis testing scheme is adopted to improve the support detection accuracy. Experimental results show that the proposed error correction algorithm has higher accuracy than the traditional algorithms.

In Chapter 5, we turn to an inference problem in gene regulatory networks. We consider models of gene regulatory networks based on chemical master equations, and study the problem of estimating stochastic rate constants therein [16]. Such models provide the most precise description of the network processes; however, they are also computationally the most demanding. We limit our focus on small-sized networks with a known structure but unknown rate constants. We approximate a chemical master equation by a related chemical Langevin equation [17], and employ a particle filter with the Markov Chain Monte Carlo move step to solve the rate estimation problem. Simulation studies demonstrate that the proposed technique outperforms previously considered methods while being computationally more efficient. Dynamic behavior of gene regulatory networks averaged over a large number of cells can be modeled by ordinary differential equations. For this scenario, we compute an approximation to the Cramer-Rao lower bound on the mean-square error of estimating reaction rates and demonstrate that, when the number of unknown parameters is small, the proposed particle filter can be nearly optimal.

Chapter 2

A particle filtering base-calling method for next-generation sequencing

2.1 Background

A widely used sequencing-by-synthesis platform, commercialized by Illumina, relies on reversible terminator chemistry. A sequencing task on the platform is preceded by the preparation of a library of single-stranded short templates created by performing random fragmentation of the target DNA sample. Each single-stranded fragment in the library is placed on a glass surface (i.e., the flow cell [18]) and subjected to bridge amplification in order to create a cluster of identical copies of DNA templates [19]. The flow cell contains eight *lanes* where each lane is divided into a hundred of nonoverlapping *tiles*. The order of nucleotides in a DNA template is identified by synthesizing its complementary strand while relying on reversible terminator chemistry [3]. Ideally, in every sequencing cycle, a single fluorescently labeled nucleotide is incorporated into the complementary strand on each copy of the template in a cluster. The incorporated nucleotide is a Watson-Crick complement of the first unpaired base of the template. In reversible terminator chemistry, four distinct fluorescent tags are used to label the four bases, and are detected by C-CD imaging technology. The acquired images are processed in order to obtain

intensity signals indicating the type of nucleotide incorporated in each cycle. These raw signal intensities are then analyzed by a base-calling algorithm to infer the order of nucleotides in each of the templates.

Quality of the acquired raw signals is adversely affected by the imperfections in the underlying sequencing-by-synthesis and signal acquisition processes. The imperfections are manifested as various sources of uncertainties. For instance, a small fraction of the strands being synthesized may fail to incorporate a base, or they may incorporate multiple bases in a single test cycle. These effects are referred to as phasing and pre-phasing, respectively, and they result in an incoherent addition of the signals generated by the synthesis of the complementary strands on the copies of the template. Other sources of uncertainty are due to cross-talk and delay effects in the optical detection process, the residual effects that are readily observed between subsequent test cycles, signal decay, and measurement noise.

Illumina's sequencing platforms are supported by a commercial base-calling algorithm called Bustard. While Bustard is computationally very efficient, its base-calling error rates can be significantly improved by various computationally more demanding schemes [20]. Such schemes include work presented in [21–24]. Among the proposed methods, the BayesCall algorithm [23] has been shown to significantly outperform Bustard in terms of the achievable base calling error rates. By relying on a full parametric model of the acquired signal, BayesCall builds a Bayesian inference framework capable of providing valuable probabilistic information that can be used in downstream

applications. However, its performance gains come at high computational costs. A modified version of the BaseCall algorithm named naiveBayesCall [24] performs base calling in a much more efficient way, but its accuracy deteriorates (albeit remains better than Bustard's). Both BayesCall and naiveBayesCall rely on expectation-maximization (EM) framework that employs a Markov chain Monte Carlo (MCMC) sampling strategy to estimate the parameters of the statistical model describing the signal acquisition process. This parameter estimation step turns out to be very time-consuming, limiting practical feasibility of the proposed schemes. Highly accurate and practically feasible parameter estimation and base-calling remain a challenge that needs to be addressed.

In this chapter, we introduce a Hidden Markov Model (HMM) representation of the acquired signals. Relying on the HMM model and particle filtering (i.e., sequential Monte Carlo) techniques, we develop a novel base calling and parameter estimation scheme and discuss some important practical aspects of the proposed method.

2.2 Contributions

In this chapter, we develop a HMM representation of the signal acquired by Illumina's sequencing-by-synthesis platforms and develop a particle filtering (i.e., sequential Monte Carlo) base-calling scheme that we refer to as ParticleCall. When relying on the BayesCall's Markov Chain Monte Carlo implementation of the EM algorithm (MCEM) to estimate system parameter-

s, ParticleCall achieves the same error rate performance as BayesCall while reducing the time needed for base calling by a factor of 3. To improve the speed of parameter estimation, we develop a particle filter implementation of the EM algorithm (PFEM). PFEM significantly reduces parameter estimation time while leading to a very minor deterioration of the accuracy of base calling. Finally, we demonstrate that ParticleCall has the best discrimination ability among all of the considered base calling schemes.

The ParticleCall software package related to the algorithm described in this chapter is freely available at https://sourceforge.net/projects/particlecall.

2.3 Mathematical model

To describe the signal acquired by the Illumina's sequencing-by-synthesis platform, a parametric model was proposed in [23]. Basic components of the model are overviewed below.

A length-L DNA template sequence is represented by a $4 \times L$ matrix S, where the i^{th} column of S, $\mathbf{s}_i$, is considered to be a randomly generated unit vector with a single non-zero entry indicating the type of the i^{th} base in the sequence. We follow the convention where the first component of the vector $\mathbf{s}_i$ corresponds to the base A, the second to C, the third to G, and the fourth to T and denote them as $\mathbf{e}_A, \mathbf{e}_C, \mathbf{e}_G, \mathbf{e}_T$. The goal of base-calling is to infer unknown S from the signals obtained by optically detecting nucleotides incorporated during the sequencing-by-synthesis process.

Let p denote the average fraction of strands that fail to extend in a test cycle. Phasing is modeled as a Bernoulli random variable with probability p. Let q denote the average fraction of strands which extend by more than one base in a single test cycle. Pre-phasing is modeled as a Bernoulli random variable with probability q. Length of the synthesized strand changes from i to j with probability

$$P_{ij} = \begin{cases} p, & \text{if } j = i, \\ 1 - p - q, & \text{if } j = i + 1, \\ q, & \text{if } j = i + 2, \\ 0, & \text{otherwise.} \end{cases}$$

Let P denote an $(L + 1) \times (L + 1)$ transition matrix with entries P_{ij} defined above, $1 \leq i, j \leq L + 1$. The signal generated over L cycles of the synthesis process is affected by phasing and pre-phasing and can be expressed as $X = SH$, where $H = (H_{i,j})$ is an $L \times L$ matrix with entries $H_{i,j} = [P^j]_{1(i+1)}$, the probability that a synthesized strand is of length i after j cycles. Here P^j denotes the j^{th} power of matrix P. The decay in signal intensities over cycles (caused by DNA loss due to primer-template melting, digestion by enzymatic impurities, DNA dissociation, misincorporation, etc.) is modeled by the per-cluster density random parameter λ_t,

$$\lambda_t = (1 - d_t)\lambda_{t-1} + (1 - d_t)\lambda_{t-1}\epsilon_t, \tag{2.1}$$

where $\epsilon_t \sim \mathcal{N}(0, \sigma_t^2)$ is a one-dimensional Gaussian random variable and d_t is the per-cluster density decay parameter within $[0, 1]$. We represent the t^{th} column of H as $\mathbf{h}_t$ and the t^{th} column of X as $\mathbf{x}_t$. Incorporating the decay

into the model, the signal generated in cycle t is expressed as

$$\mathbf{x}_t = \lambda_t S \mathbf{h}_t, \tag{2.2}$$

where $\mathbf{x}_t = \left[x_t^A \; x_t^C \; x_t^G \; x_t^T\right]'$ is the vector of signals generated in each of the optical channels. Assuming Gaussian observation noise, the measured intensities at cycle t are given by

$$\mathbf{y}_t = K_t \mathbf{x}_t + \sum_{b \in \{A,C,G,T\}} x_t^b \eta_t^b, \tag{2.3}$$

where K_t denotes the 4×4 crosstalk matrix describing overlap of the emission spectra of the four fluorescent tags, and η_t^A, η_t^C, η_t^G, η_t^T are independent, identically distributed (i.i.d.) 4×1 Gaussian random vectors with zero mean and a common 4×4 covariance matrix Σ_t.

Note that, due to typically small values of p and q, the components of the vector $\mathbf{h}_t$ around its t^{th} entry are significantly greater than the remaining ones. This observation can be used to simplify the expressions (2.2) and (2.3). In particular, let $\mathbf{h}_t^w$ denote the vector obtained by windowing $\mathbf{h}_t$ around its t^{th} entry, i.e., by setting small components of $\mathbf{h}_t$ to 0. In general, we consider $l + r + 1$ dominant components of $\mathbf{h}_t$ centered at position t, $H_{t-l,t}, H_{t-l+1,t}, \ldots, H_{t,t}, \ldots H_{t+r-1,t}, H_{t+r,t}$, expression (2.2) becomes

$$\mathbf{x}_t \approx \mathbf{x}_t^w = \lambda_t S \mathbf{h}_t^w = \lambda_t \sum_{i=-l}^{r} \mathbf{s}_{t+i} H_{t+i,t}. \tag{2.4}$$

Finally, note that the signal measured in cycle t is empirically observed to contain residual effect from the previous cycle. The residual effect is mod-

eled by adding $\alpha_t(1-d_t)\mathbf{y}_{t-1}$ to $\mathbf{y}_t$, where the unknown parameter $\alpha_t \in (0,1)$. Therefore, the model can be summarized as

$$\begin{aligned}
\lambda_t|\lambda_{t-1} \sim &\ \mathcal{N}((1-d_t)\lambda_{t-1}, (1-d_t)^2\lambda_{t-1}^2\sigma_t^2), \\
\mathbf{y}_t|\mathbf{y}_{t-1}, S, \lambda_t \sim &\ \mathcal{N}(K_t\mathbf{x}_t^w + \alpha_t(1-d_t)\mathbf{y}_{t-1}, \|\mathbf{x}_t^w\|_2^2\Sigma_t), \\
\mathbf{s}_t \sim &\ \mathbf{Unif}(\mathbf{e}_A, \mathbf{e}_C, \mathbf{e}_G, \mathbf{e}_T), \\
x_t^w = &\ \lambda_t S\mathbf{h}_t^w \\
&\ t = 1, 2, \ldots, L
\end{aligned}$$

where $\|\cdot\|_2$ denotes the l_2-norm of its argument, and where $\mathbf{y}_0 = \mathbf{0}$, $\lambda_0 = 1$.

2.4 Hidden Markov Model of DNA base-calling

In this section, we reformulate the statistical description of the signal acquired by the Illumina's sequencing-by-synthesis platform as a Hidden Markov Model (HMM) [35]. HMMs comprise a family of probabilistic graphical models which describe a series of observations by a "hidden" stochastic process and are generally suitable for representing time series data. Sequencing data obtained from the Illumina's platform is a set of time-series intensities $\mathbf{y}_{1:L}$, motivating the HMM representation. HMMs provide a convenient framework for state and parameter estimation, which we exploit to develop a particle filter base-calling scheme in the next section.

For the sake of convenience, we remove the dependency between subsequent observations $\mathbf{y}_{t-1}$ and $\mathbf{y}_t$ by defining $\mathbf{y}_t^{'} = \mathbf{y}_t - \alpha_t(1-d_t)\mathbf{y}_{t-1}$, $t = 1, 2, \ldots, L$. Therefore, we can write

$$\mathbf{y}_t^{'}|S, \lambda_t \sim \mathcal{N}(K_t\mathbf{x}_t^w, \|\mathbf{x}_t^w\|_2^2\Sigma_t). \tag{2.5}$$

Components of $\mathbf{y}'_{1:L}$ are the observations of our HMM, and depend on the underlying signals $\mathbf{x}_{1:L}$. Moreover, let S_t^w denote the $4 \times (l + r + 1)$ windowed submatrix of S, i.e.,

$$S_t^w = [\mathbf{s}_{t-l}\ \mathbf{s}_{t-l+1}\ \ldots\ \mathbf{s}_t\ \ldots\ \mathbf{s}_{t+r}].$$

Since $\mathbf{x}_t^w = \lambda_t S \mathbf{h}_t^w = \lambda_t \sum_{i=t-l}^{t+r} \mathbf{s}_i H_{i,t}$, it is clear that $\mathbf{y}'_t$ depends on λ_t and S_t^w. Therefore, we define the state of the HMM to be the combination of λ_t and S_t^w – the per-cluster density at cycle t and the collection of $(l + r + 1)$ bases around (and including) the base in position t, respectively.

The proposed HMM representation is illustrated in Fig 2.4. The observation dynamics that characterize the relationship between $\mathbf{y}'_t$ and the hidden states (S_t^w, λ_t) are given by the distribution $g(\mathbf{y}'_t | S_t^w, \lambda_t)$. It is straightforward to show from (2.5) that

$$g(\mathbf{y}'_t | S_t^w, \lambda_t) \sim \mathcal{N}(K_t \mathbf{x}_t^w, \|\mathbf{x}_t^w\|_2^2 \Sigma_t). \tag{2.6}$$

On the other hand, the state transition dynamics is described by the transition probability between subsequent states, $(S_{t-1}^w, \lambda_{t-1})$ and (S_t^w, λ_t). Since S_t^w and λ_t are independent, the transition probability is

$$f(S_t^w, \lambda_t | S_{t-1}^w, \lambda_{t-1}) = f_1(S_t^w | S_{t-1}^w) f_2(\lambda_t | \lambda_{t-1}). \tag{2.7}$$

The second term on the right-hand side of (2.7), $f_2(\lambda_t | \lambda_{t-1})$, is known from the density decay model (2.1),

$$f_2(\lambda_t | \lambda_{t-1}) \sim \mathcal{N}((1 - d_t)\lambda_{t-1}, (1 - d_t)^2 \lambda_{t-1}^2 \sigma_t^2).$$

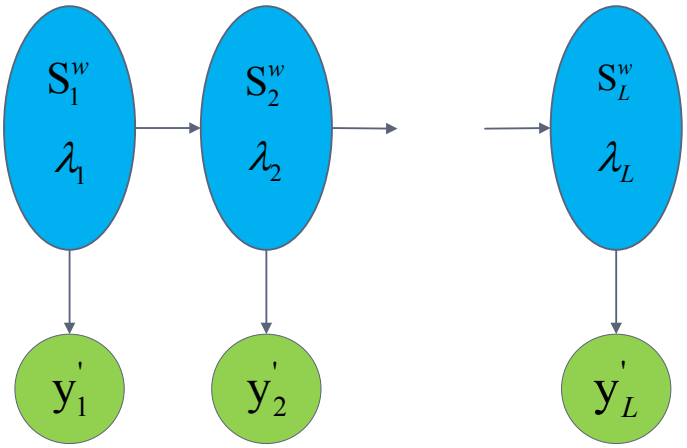

Figure 2.1: *A hidden Markov model of the generated signal in Illumina sequencing-by-synthesis platforms.*

For notational convenience, we use $\mathbf{s}_{t,1}^w, \ldots, \mathbf{s}_{t,l+r+1}^w$ to denote the set of $l+r+1$ column vectors of S_t^w. Note that for $k = 2, 3, \ldots, l+r+1$, the column vectors $\mathbf{s}_{t-1,k}^w$ in S_{t-1}^w and the column vectors $\mathbf{s}_{t,k-1}^w$ in S_t^w actually represent the same base. Therefore, the transition model between them can be represent by a δ function as

$$\begin{aligned} p(\mathbf{s}_{t,k-1}^w | \mathbf{s}_{t-1,k}^w) &= \delta_{\{\mathbf{s}_{t,k-1}^w = \mathbf{s}_{t-1,k}^w\}} \\ &= \begin{cases} 1, & \text{if } \mathbf{s}_{t,k-1}^w = \mathbf{s}_{t-1,k}^w, \\ 0, & \text{if } \mathbf{s}_{t,k-1}^w \neq \mathbf{s}_{t-1,k}^w. \end{cases} \end{aligned}$$

Let $U(\{\mathbf{e}_A, \mathbf{e}_C, \mathbf{e}_G, \mathbf{e}_T\})$ denote a uniform distribution on the support set of unit vectors $(\{\mathbf{e}_A, \mathbf{e}_C, \mathbf{e}_G, \mathbf{e}_T\})$. We assume no correlation between consecutive bases of the template sequence, i.e., $\mathbf{s}_{t,l+r+1}^w$ is generated from $U(\{\mathbf{e}_A, \mathbf{e}_C, \mathbf{e}_G, \mathbf{e}_T\})$.

Therefore, $f_1(S_t^w|S_{t-1}^w)$ can be written as

$$f_1(S_t^w|S_{t-1}^w) = \left(\prod_{k=2}^{l+r+1} \delta_{\{\mathbf{s}_{t-1,k}^w = \mathbf{s}_{t,k-1}^w\}} \right) u(\mathbf{s}_{t,l+r+1}^w),$$

where $u(\cdot) \sim U(\{\mathbf{e}_A, \mathbf{e}_C, \mathbf{e}_G, \mathbf{e}_T\})$. Hereby, all the components of the HMM are specified.

2.5 ParticleCall base-calling algorithm

The goal of base calling is to determine the order of nucleotides in a template from the acquired signal $\mathbf{y}_{1:t}$. This can be rephrased as the problem of inferring the most likely sequence of states (S_t^w, λ_t) of the HMM in (2.6)-(2.7) from the observed sequence $\mathbf{y}'_{1:t}$ (clearly, $\mathbf{s}_{1:L}$ follows directly from S_t^w). We assume that the parameters $\Theta = \{p, q, d_{1:L}, \alpha_{1:L}, \sigma_{1:L}, K_{1:L}, \Sigma_{1:L}\}$ are common for all clusters within a tile, and that they are provided by a parameter estimation step discussed in the following section. In this section, we introduce a novel base calling algorithm ParticleCall which relies on particle filtering techniques to sequentially infer (S_t^w, λ_t) and, therefore, recover the matrix S.

In general, particle filtering (i.e., sequential Monte Carlo) methods generate a set of particles with associated weights to estimate the posteriori distribution of unknown variables given the acquired measurements [34]. In the proposed HMM framework, we sequentially calculate the posteriori distribution of the columns of S, $p(\mathbf{s}_t|\mathbf{y}'_{1:t}), t = 1, 2, \ldots, L$, and find the maximum a posteriori (MAP) estimates of $\mathbf{s}_t$ by solving

$$\hat{\mathbf{s}}_t = \arg \max_{\mathbf{s}_t \in \{\mathbf{e}_A, \mathbf{e}_C, \mathbf{e}_G, \mathbf{e}_T\}} \{p(\mathbf{s}_t|\mathbf{y}'_{1:t})\}.$$

Our algorithm relies on a sequential importance sampling/resampling (SISR) particle filter scheme [25] to calculate $p(S_t^w, \lambda_t | y'_{1:t})$. Different choices and approximation methods of proposal densities are considered in [26–30]. We directly use the transition (2.7) as the proposal density. This sequential importance sampling suffers from degeneracy and the variance of the importance weights will increase over time. To address the degeneracy problem, a resampling step is introduced in order to eliminate samples which have small normalized importance weights. Common resampling methods include multinomial resampling [25], residual resampling [31] and systematic resampling [32, 33]. We measure degeneracy of the algorithm using the effective sample size K_{eff} and, for the sake of simplicity, employ multinomial resampling strategy. If we denote the number of particles by N_p, then $K_{\text{eff}} = (\sum_{k=1}^{N_p} (w_t^{(i)})^2)^{-1}$ and resampling step is used when K_{eff} is below a fixed threshold $N_{threshold}$. $N_{threshold}$ of size $O(N_p)$ is typically sufficient [25]. In our implementation, we set $N_{threshold} = N_p/2$.

We omit further details for brevity and formalize the ParticleCall algorithm below.

Algorithm 1 ParticleCall base-calling algorithm

1. Initialization:

1.1 Initialize particles:

for $i = 1 \to N_p$ **do**

Sample each column of the submatrix $S_1^{w,(i)}$ from $U(\{\mathbf{e}_A, \mathbf{e}_C, \mathbf{e}_G, \mathbf{e}_T\})$;

Sample $\lambda_1^{(i)}$ from a Gaussian distribution with mean 1, and the variance calculated using Bustard's estimates of λ in the first 10 test cycles.

end for

1.2 Compute and normalize weights for each particle according to $w_1^{(i)} \propto g(\mathbf{y}'_1|S_1^{w,(i)}, \lambda_1^{(i)})$.

2. Run iteration $t(t \geq 2)$:

2.1 Sampling:

for $i = 1 \to N_p$ **do**

Sample $S_t^{w,(i)}, \lambda_t^{(i)} \sim f(\cdot, \cdot|S_{t-1}^{w,(i)}, \lambda_{t-1}^{(i)})$ according to (2.7).

end for

2.2 Update the importance weight

$$w_t^{(i)} \propto w_{t-1}^{(i)} g(\mathbf{y}'_t|S_t^{w,(i)}, \lambda_t^{(i)}).$$

2.3 Normalize the weights. Calculate the posteriori probability of $\mathbf{s}_t$ and obtain the estimate $\hat{\mathbf{s}}_t$.

2.4 Resampling:

if $K_{\text{eff}} = \left(\sum_{k=1}^{N_p} (w_t^{(i)})^2\right)^{-1} \leq N_{threshold}$ **then**

Draw N_p samples $\{\bar{S}_t^{w,(j)}, \bar{\lambda}_t^{(j)}, j = 1, \ldots, N_p\}$ from $\{S_t^{w,(i)}, \lambda_t^{(i)}, i = 1, \ldots, N_p\}$ with probabilities proportional to $\{w_t^{(i)}, i = 1, \ldots, N_p\}$.

Assign equal weight to each particle, $\bar{w}_t^{(i)} = 1/N_p$.

end if

Since S_t^w in the HMM states are discrete with a finite alphabet, and the transitions of S_t^w and λ_t are independent according to (2.7), it is possible to Rao-Blackwellize the ParticleCall algorithm. Rao-Blackwellization is used to marginalize part of the states in the particle filter, hence reducing the number of needed particles N_p [27]. We marginalize the discrete states S_t^w and

reduce the hidden process to λ_t, while relying on the particle filter to calculate $p(\lambda_{1:t}|\mathbf{y}'_{1:t})$.

The original posterior distribution of the states can be expressed as

$$p(\lambda_{1:t}, S^w_{1:t}|\mathbf{y}'_{1:t}) = p(S^w_{1:t}|\mathbf{y}'_{1:t}, \lambda_{1:t})p(\lambda_{1:t}|\mathbf{y}'_{1:t}).$$

Since $p(\lambda_{1:t}|\mathbf{y}'_{1:t}) \propto p(\mathbf{y}'_t|\mathbf{y}'_{1:t-1}, \lambda_{1:t})p(\lambda_t|\lambda^{(i)}_{t-1})$, where $\lambda^{(i)}_{t-1}$ is a sample from $p(\lambda_{1:t-1}|\mathbf{y}'_{1:t-1})$, we can state the Rao-Blackwellized ParticleCall algorithm as below.

Algorithm 2 Rao-Blackwellized ParticleCall algorithm

1. Initialization:
1.1 Initialize particles:
for $i = 1 \to N_p$ **do**
  Sample $\lambda^{(i)}_1$ from a Gaussian distribution with mean 1, and the variance calculated using Bustard's estimates of λ in the first 10 test cycles.
end for
1.2 Compute and normalize weights for each particle according to $w^{(i)}_1 \propto g(\mathbf{y}'_1|\lambda^{(i)}_1) \propto \sum_{S^w_1} g(\mathbf{y}'_1|S^w_1, \lambda^{(i)}_1)$.
1.3 Calculate the discrete distribution $p(S^w_1|\mathbf{y}_1, \lambda^{(i)}_1)$ for each i.
2. Run iteration $t(t \geq 2)$:
2.1 Sampling:
for $i = 1 \to N_p$ **do**
  Sample $\lambda^{(i)}_t \sim f(\cdot|\lambda^{(i)}_{t-1})$.
end for
2.2 Update the importance weight $w^{(i)}_t \propto w^{(i)}_{t-1} g(\mathbf{y}'_t|\mathbf{y}'_{1:t-1}, \lambda^{(i)}_{1:t})$. and normalize the weights.
2.3 Resample if $K_{\text{eff}} \leq N_{threshold}$
2.4 Update $p(S^w_t|\mathbf{y}'_{1:t}, \lambda^{(i)}_{1:t})$
for $i = 1 \to N_p$ **do**
  Update $p(S^w_t|\mathbf{y}'_{1:t}, \lambda^{(i)}_{1:t})$ using $p(S^w_{t-1}|\mathbf{y}'_{1:t-1}, \lambda^{(i)}_{1:t-1})$ and $\lambda^{(i)}_t$.
end for

In step 2.2 of Algorithm 2, the quantity $g(\mathbf{y}'_t|\mathbf{y}'_{1:t-1}, \lambda^{(i)}_{1:t})$ can be obtained by marginalizing over discrete states S^w_t and S^w_{t-1},

$$\begin{aligned} g(\mathbf{y}'_t|\mathbf{y}'_{1:t-1}, \lambda^{(i)}_{1:t}) &= \sum_{S^w_t} p(\mathbf{y}'_t|\mathbf{y}'_{1:t-1}, S^w_t, \lambda^{(i)}_{1:t}) p(S^w_t|\mathbf{y}'_{1:t-1}, \lambda^{(i)}_{1:t}) \\ &= \sum_{S^w_t} p(\mathbf{y}'_t|S^w_t, \lambda^{(i)}_t) \sum_{S^w_{t-1}} \big[p(S^w_t|S^w_{t-1}, \mathbf{y}'_{1:t-1}, \lambda^{(i)}_{1:t}) \times \\ &\quad p(S^w_{t-1}|\mathbf{y}'_{1:t-1}, \lambda^{(i)}_{1:t}) \big], \end{aligned} \tag{2.8}$$

where $p(\mathbf{y}'_t|S^w_t, \lambda^{(i)}_t)$ is the observation density,

$$p(S^w_{t-1}|\mathbf{y}'_{1:t-1}, \lambda^{(i)}_{1:t}) = p(S^w_{t-1}|\mathbf{y}'_{1:t-1}, \lambda^{(i)}_{1:t-1}).$$

Due to the independence of the state transitions, and $p(S^w_t|S^w_{t-1}, \mathbf{y}'_{1:t-1}, \lambda^{(i)}_{1:t}) = p(S^w_t|S^w_{t-1})$ due to the Markov property and the independence of the state transitions.

In step 2.4 of Algorithm 2, the update equation is obtained as

$$\begin{aligned} p(S^w_t|\mathbf{y}'_{1:t}, \lambda^{(i)}_{1:t}) &\propto p(S^w_t, \mathbf{y}'_t, \lambda^{(i)}_t|\mathbf{y}'_{1:t-1}, \lambda^{(i)}_{1:t-1}) \\ &= \sum_{S^w_{t-1}} p(\mathbf{y}'_t, S^w_t, \lambda^{(i)}_t|\mathbf{y}'_{1:t-1}, S^w_{t-1}, \lambda^{(i)}_{1:t-1}) p(S^w_{t-1}|\mathbf{y}'_{1:t-1}, \lambda^{(i)}_{1:t-1}) \\ &= \sum_{S^w_{t-1}} \big[p(\mathbf{y}'_t|S^w_t, \lambda^{(i)}_t, \mathbf{y}'_{1:t-1}, S^w_{t-1}, \lambda^{(i)}_{1:t-1}) \times \\ &\quad p(S^w_t, \lambda^{(i)}_t|\mathbf{y}'_{1:t-1}, S^w_{t-1}, \lambda^{(i)}_{1:t-1}) p(S^w_{t-1}|\mathbf{y}'_{1:t-1}, \lambda^{(i)}_{1:t-1}) \big] \\ &= p(\mathbf{y}'_t|S^w_t, \lambda^{(i)}_t) p(\lambda^{(i)}_t|\lambda^{(i)}_{t-1}) \times \\ &\quad \sum_{S^w_{t-1}} p(S^w_t|S^w_{t-1}) p(S^w_{t-1}|\mathbf{y}'_{1:t-1}, \lambda^{(i)}_{1:t-1}) \end{aligned} \tag{2.9}$$

2.6 Parameter estimation

To determine the set of parameters Θ needed to run the proposed ParticleCall base calling algorithm, one could rely on the MCMC implementation of the EM algorithm (MCEM) proposed in [23]. In section Results, we demonstrate the performance of the ParticleCall algorithm that relies on the MCEM parameter estimation scheme. Note, however, that the MCMC sampling strategy employed by MCEM requires a lengthy burn-in period and a very large sample size to perform the expectation step. Therefore, the MCEM parameter estimation scheme is computationally rather intensive and requires significant computational resources if it is to be used for processing large sequencing data sets. As an alternative, we develop an EM parameter estimation scheme which relies on the proposed HMM and uses samples generated by a particle filter to evaluate the expectation of the likelihood function. We refer to this algorithm as the particle filter EM (PFEM). The speed and accuracy of the proposed scheme is practically sound for use in next generation sequencing platforms.

2.6.1 Assumptions on parameters

Recall that the set of parameters needed to be estimated before running ParticleCall is $\Theta = \{p, q, d_{1:L}, \alpha_{1:L}, \sigma_{1:L}, K_{1:L}, \Sigma_{1:L}\}$. The phasing and prephasing parameters p and q are assumed to be the same for each sequencing lane and are estimated using the same procedure as Bustard (see, e.g., [23]). The remaining parameters are assumed to be cycle-dependent and need to be estimated for each tile. The cycle-dependency assumption on the parameters

can lead to a substantial improvement in the base-calling accuracy [20]. In order to avoid over-fitting, we assume that parameters remain constant within a short window of cycles and then change to a different set of values. To track the changes in the parameters, we first divide the total read length L into several non-overlapping windows and then perform our parameter estimation window-by-window. To further reduce the number of parameters and improve the estimation efficiency, we assume that the parameters $d_{1:L}$ and $\sigma_{1:L}$ are uniformly distributed over an interval and incorporate them into the hidden states of the HMM model. Therefore, only the mean and variance of these parameters, i.e., d_{mean}, d_{var}, σ_{mean}, and σ_{var} need to be estimated. Computational results demonstrate that these two assumptions does not affect the accuracy of base-calling.

2.6.2 Particle filter EM algorithm

In the early sequencing cycles, effects of phasing and prephasing are relatively small. Therefore, we may ignore phasing and prephasing to facilitate straight-forward computation of the initial estimates of the remaining parameters. In particular, the signal generated in the early cycles t is approximated as

$$\mathbf{x}_t = \lambda_t \mathbf{s}_t. \tag{2.10}$$

Replacing (2.2) by (2.10) leads to a simplified model that allows for straightforward base calling and inference of the parameters by means of linear regression. We use these values to obtain the estimates of d_{mean}, d_{var}, σ_{mean}, and σ_{var},

and to initialize the remaining parameters α, K, Σ, in the particle filter EM parameter estimation procedure.

The parameter estimation is performed window-by-window and is conducted using n reads randomly chosen from a tile (in our experiments, we use $n = 200$). Assume the window length is w, and denote the window index by m. The particle filter EM (PFEM) algorithm finds parameters for one window and then uses these values to initialize the search for parameters in the next window. We illustrate the procedure for the first window here (the same procedure is repeated in the following windows). Let $\Theta_1^i = \{\alpha^i, K^i, \Sigma^i\}$ denote the set of parameters for window 1 in the ith iteration of the EM scheme. The estimate of Θ_1^i is given by

$$\Theta_1^i = \arg\max_{\Theta_1} L_1(\Theta_1^{i-1}), \tag{2.11}$$

where $L_1(\Theta_1^{i-1}) = \sum_{j=1}^{n} L_{1,j}(\Theta_1^{i-1})$ is the sum of the log-likelihood functions over the reads in the training set. The log-likelihood function for each read, $L_{1,j}(\Theta_1^{i-1})$, is obtained as

$$L_{1,j}(\Theta_1^{i-1}) = \log P(\mathbf{y}_{1:w}|\Theta_1^{i-1}) = \mathbf{E}\left[\log P(\mathbf{y}_{1:w}, \mathbf{s}_{1:w}, \lambda_{1:w}|\Theta_1^{i-1})\right], \tag{2.12}$$

where the expectation is taken with respect to $P(\mathbf{s}_{1:w}, \lambda_{1:w}|\mathbf{y}_{1:w}, \Theta_1^{i-1})$. We rely on an SISR particle filtering scheme to generate equally weighted sample trajectories from $P(\mathbf{s}_{1:w}, \lambda_{1:w}|\mathbf{y}_{1:w}, \Theta_1^{i-1})$. Based on (2.6) and (2.7), we calculate $\log P(\mathbf{y}_{1:w}, \mathbf{s}_{1:w}, \lambda_{1:w}|\Theta_1^{i-1})$ for these samples and compute their average to approximate the expectation in (2.12). The maximization (2.11) is performed

by solving equations obtained after taking gradients of $L_1(\Theta_1^{i-1})$ over the parameters and setting them to 0. In our experiment, the PFEM parameter estimation scheme performs 30 EM iterations and uses 600 samples from the particle filter for each window.

2.7 Results

The proposed method is evaluated on a data set obtained by sequencing phiX174 bacteriophage using Illumina Genome Analyzer II with the cycle length 76. This is a short genome with a known sequence which enables reliable performance comparison of different base-calling techniques. We tested ParticleCall and several other algorithms on a tile containing 77337 reads, and present the results here. All the codes are written in C and the tests are run on a desktop with an Intel Core i7 4-core 3GHz processor.

2.7.1 Performance of ParticleCall

The base calling error rates are computed by aligning the reads to the reference genome and evaluating frequency of mismatches. Reads that could not be aligned to the reference with at least 70% matches are discarded. Note that the error rates and speed of the proposed ParticleCall algorithm and the parameter estimation scheme are affected by the parameters l, r, particle number N_p, and parameter estimation window length w. We ran ParticleCall with $l = r \in \{1, 2, 4\}$. Increasing l and r beyond $l = r = 1$ did not affect the performance while it significantly slowed down the algorithm.

This is due to small values of the phasing and prephasing probabilities, which are estimated to be $p = 3.54 \times 10^{-8}$ and $q = 0.00335$. Therefore, in the remainder of the chapter, we set $l = r = 1$. The accuracy of base-calling for different N_p is shown in Table 2.7.1. As seen there, for the original ParticleCall algorithm, $N_p = 800$ leads to high performance with reasonable speed. Rao-Blackwellized ParticleCall can achieve the same accuracy with fewer particles (in particular, $N_p = 300$); however, its effective running time is 3 times that of the original ParticleCall with the same performance. This is because the Rao-Blackwellization steps in (2.8) and (2.9) require evaluating a sum over all possible S_t^w ($4^3 = 64$ for our choice $l = r = 1$), resulting in a fairly large number of basic operations needed to calculate exact distribution of the discrete variables. Therefore, for further performance comparisons, we rely on the original ParticleCall algorithm (formalized as Algorithm 1). Table 2.2 shows the ParticleCall base calling error rate and parameter estimation times for different window lengths w. In the remainder of the chapter, we set $w = 5$ as it leads to desirable performance/speed characteristics of the algorithm.

2.7.2 Performance comparison of different algorithms

The error rates and speed of the proposed ParticleCall algorithm are compared with those of BayesCall, naiveBayesCall, Rolexa, and Bustard. We run ParticleCall both with parameters provided by the computationally intensive MCEM algorithm as well as with those inferred by the PFEM parameter estimation scheme proposed in this chapter. The results are reported in Ta-

Table 2.1: Comparison of ParticleCall with different N_p

Method	N_p	error rate	base-calling time (min)
ParticleCall (via MCEM)	400	0.0126	46
	800	0.0124	88
	1200	0.0124	130
ParticleCall (via PFEM)	400	0.0128	46
	800	0.0125	91
	1200	0.0125	133
Rao-Blackwellized ParticleCall (via MCEM)	100	0.0128	103
	200	0.0125	190
	300	0.0124	287
	400	0.0124	386

Table 2.2: ParticleCall parameter estimation with different w

Window length w	base-calling error rate error rate	parameter estimation time (min)
4	0.0125	50
5	0.0125	39
6	0.0127	29
7	0.0130	25

Table 2.3: Comparison of error rates and speed

Method	error rate	base-calling time (min)	parameter estimation time (min)
Bustard	0.0152	2 (total)	
Rolexa	0.0170	35 (total)	
naiveBayesCall	0.0132	21	1139
BayesCall	0.0124	231	1139
ParticleCall (via MCEM)	0.0124	88	1139
ParticleCall (via PFEM)	0.0125	91	39

ble 3. Note that Rolexa generally outputs the so-called IUPAC codes, unlike all the other considered algorithms which provide sequences of nucleotides A, C, G, and T. To allow a comparison, we enforce Rolexa to output sequences of nucleotides as well. The comparison of per-cycle error rates is shown in Fig 2.7.2. The standard deviation of the error rates is shown in Fig 2.7.2.

It can be seen from Table 2.3 and Fig 2.7.2 that ParticleCall, BayesCall and naiveBayesCall all have improved base-calling accuracy compared to Bustard. BayesCall is highly accurate but relatively slow – it requires approximately 4 hours to complete base-calling for one tile of the data. naiveBayesCall significantly improves base-calling speed over BayesCall but it does so at the expense of incurring higher error rate. Our ParticleCall base-calling algorithm has the same accuracy as BayesCall, while being roughly 3 times faster. Fig 2.7.2 shows that both ParticleCall and BayesCall are more accurate than naiveBayesCall in the early cycles and improve over Bustard in all cycles. Note that Bustard outperforms Rolexa, which is consistent with the results in [20]. Moreover, we see from Table 2.3 that performing parameter estimation via the MCEM algorithm proposed in [23] requires 19 hours, while the particle filter implementation of the EM estimation scheme proposed in this chapter takes only 39 minutes. As evident from Table 2.3, running ParticleCall with parameters obtained by the PFEM scheme leads to only a minor performance degradation compared to running it with parameters obtained by the MCEM algorithm. Running ParticleCall base calling along with the PFEM parameter estimation scheme takes about 2 hours per tile, which is 9 times faster than

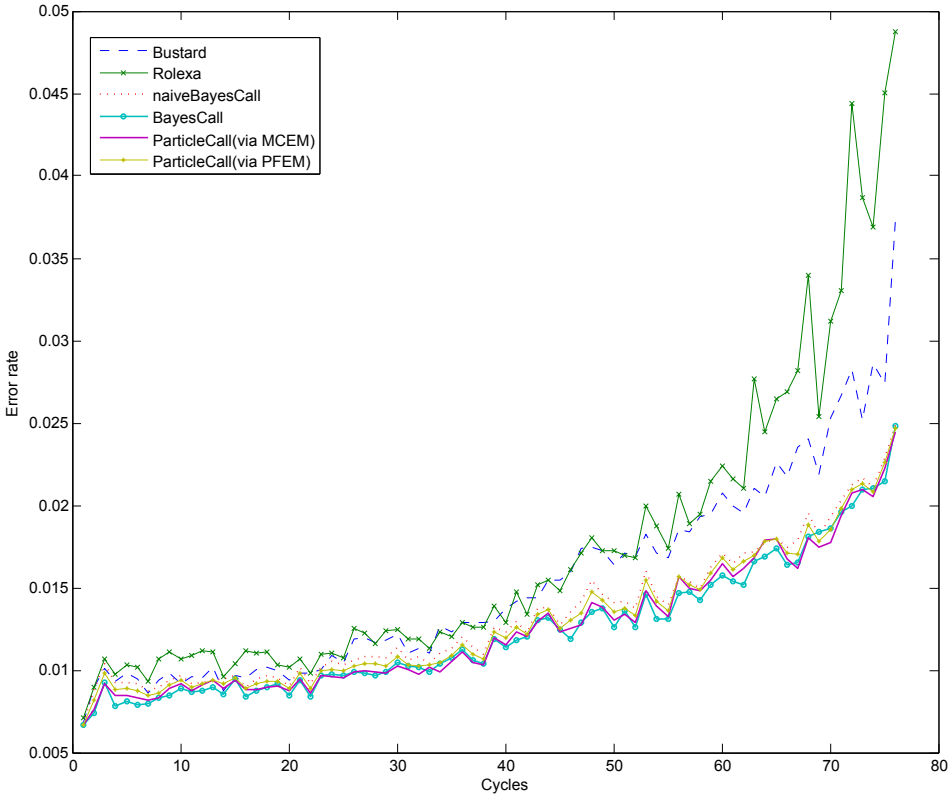

Figure 2.2: *Per-cycle error rates of ParticleCall, BayesCall, naiveBayesCall, Rolexa and Bustard.*

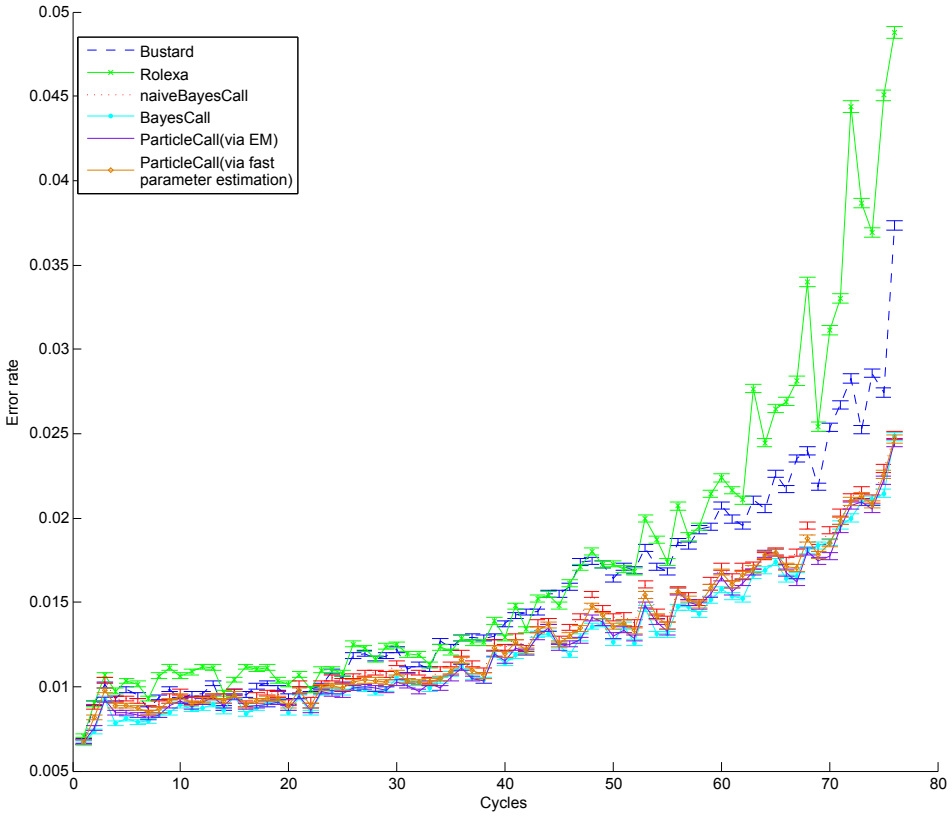

Figure 2.3: *Standard deviation of error rates of ParticleCall, BayesCall, naiveBayesCall, Rolexa and Bustard.*

the total time required by the less accurate naiveBayesCall.

2.7.3 Quality scores

Quality scores are used to characterize confidence in the outcome of the base-calling procedures. They are computed as part of the analysis of the acquired raw data and may be used to filter out reads of suspect quality, or to shorten the reads if the quality scores of individual bases fall below certain thresholds. They can also provide confidence information for downstream analysis including sequence assembly and SNP and genotype calling. Frequently used are the so-called *phred* quality scores, which were originally developed to assess the quality of the conventional Sanger sequencing and automate large-scale sequencing projects. Phred scores are also often provided by the algorithms used for base-calling in next generation sequencing platforms. Formally, the phred score for a called base in the cycle t, $\hat{\mathbf{s}}_t$, is defined as

$$Q_{phred}(\hat{\mathbf{s}}_t) = -10 \log_{10} P(\hat{\mathbf{s}}_t \neq \mathbf{s}_t).$$

Essentially, $Q_{phred}(\hat{\mathbf{s}}_t)$ is the scaled logarithm of the error probability. Higher quality scores imply smaller probability of the base-calling error. For the proposed ParticleCall algorithm, probability of correctly calling a base can be obtained from the posteriori probability as

$$P(\hat{\mathbf{s}}_t \neq \mathbf{s}_t) = 1 - p(\mathbf{s}_t|\mathbf{y}'_{1:t}).$$

Quality scores can be used to compare the discrimination ability of different algorithms. The discrimination score $D(\epsilon)$ at error tolerance ϵ is defined as

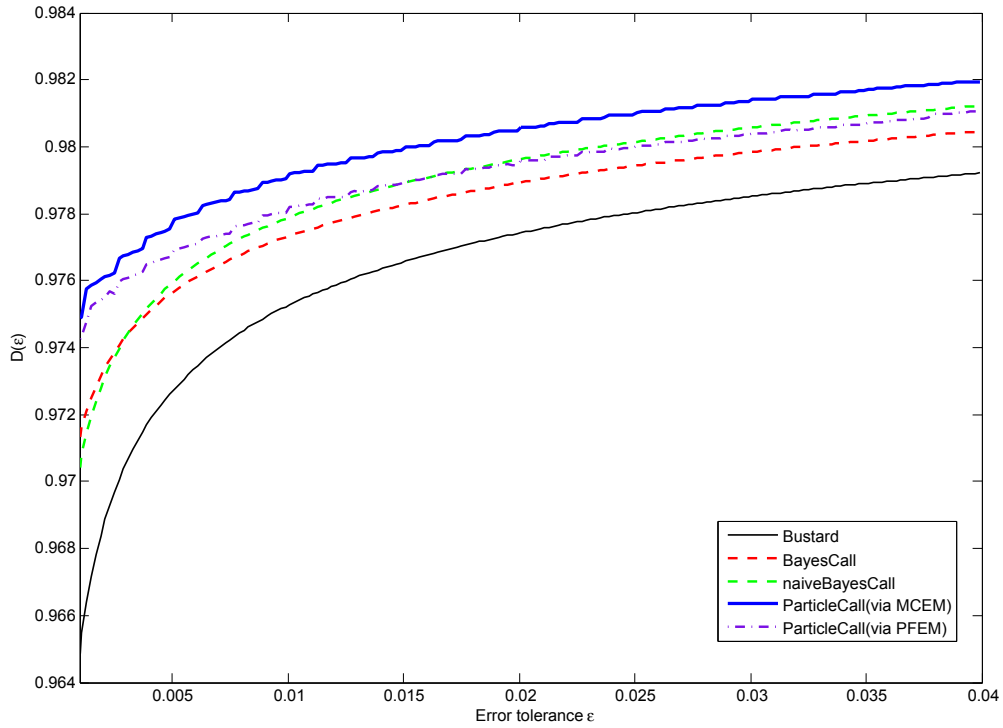

Figure 2.4: *Discrimination ability $D(\epsilon)$ of quality scores vs error tolerance.*

the ratio of the correctly called bases having $P(\hat{\mathbf{s}}_t \neq \mathbf{s}_t) < \epsilon$ (i.e., the quality score higher than $-10\log_{10}(\epsilon)$) to all called bases. Fig 2.7.3 compares the discrimination ability of ParticleCall, BayesCall, naiveBayesCall and Bustard. It shows that for a reasonable error tolerance ϵ, ParticleCall with parameters obtained through MCEM has better discrimination ability than BayesCall, naiveBayesCall and Bustard, while ParticleCall with parameters obtained through PFEM has discrimination ability close to naiveBayesCall and better

than other algorithms. In other words, when a small cutoff error tolerance ϵ is set and all the bases with quality scores below ϵ are considered invalid, ParticleCall provides the most accurate results among the considered base-calling schemes.

2.7.4 Effects of improved base-calling accuracy on de novo sequence assembly

In shotgun sequencing, a long target sequence is oversampled by a library of randomly fragmented copies of the target, and the overlaps between short reads obtained by a high-throughput platform are used to assemble the target. In *de novo* assembly, the target is reconstructed without consulting any reference [7, 8]. Performance of assembly algorithms highly depends on the accuracy of base calling. To demonstrate the effects of base-calling accuracy on assembly, we apply the Velvet assembly algorithm [8] on reads provided by Bustard, Rolexa, naiveBayesCall, BayesCall, and ParticleCall. In particular, we randomly subsample the set of reads provided by each of the base calling algorithms to emulate 5X, 10X, 15X, and 20X coverage. Then we run Velvet on each of the subsets, and evaluate commonly used metrics that quantify the quality of the assembly procedure. Specifically, we evaluate the maximum contig length and the N50 contig length. The described procedure is repeated 200 times to obtain average values of these two quality metrics. The results are shown in Table 2.4. As can be seen there, ParticleCall provides the largest N50 and maximum contig length among all of the considered base calling schemes, for all of the considered coverages.

Table 2.4: de novo assembly results

Coverage	Bustard		Rolexa		naiveBayesCall	
	N50	Max	N50	Max	N50	Max
5X	271	607	259	565	278	604
10X	1169	1750	971	1557	1180	1731
15X	3624	3823	2885	3170	3726	3908
20X	4694	4744	4529	4614	4756	4816
Coverage	BayesCall		ParticleCall via MCEM		ParticleCall via PFEM	
	N50	Max	N50	Max	N50	Max
5X	292	629	299	637	289	632
10X	1269	1831	1316	1900	1341	1865
15X	3466	3741	3742	3935	3697	3918
20X	4827	4875	5102	5116	4795	5039

2.8 Conclusions

In this chapter we proposed ParticleCall, a particle filtering algorithm for base calling in the Illumina's sequencing-by-synthesis platform. The algorithm is developed by relying on an HMM representation of the sequencing process. Experimental results demonstrate that the ParticleCall base calling algorithm is more accurate than Bustard, Rolexa, and naiveBayesCall. It is as accurate as BayesCall while being significantly faster. Quality score analysis of the reads indicates that ParticleCall has better discrimination ability than BayesCall, naiveBayesCall and Bustard. Moreover, a novel particle filter EM (PFEM) parameter estimation scheme, much faster than the existing Monte Carlo implementation of the EM algorithm, was proposed. When relying on the PFEM scheme, ParticleCall has near-optimal performance while needing much shorter total parameter estimation and base calling time.

Chapter 3

Iterative Learning for Reference-Guided DNA Sequence Assembly

3.1 Background

In next-generation sequencing, the short reads generated by a sequencing instrument are used to assemble the target genome. The assembly may be performed with or without referring to a previously determined sequence related to the target (genome, transcriptome, proteins). *De novo* assembly refers to a scenario where the reconstruction is performed without a reference sequence. This is a computationally challenging task, difficult due to the presence of perfect repeat regions in the target sequence and short lengths of the reads [38], [39]. In re-sequencing projects where the goal, for instance, may be to study genetic variations among individuals or to discover new strains of bacteria [36], [40], a reference is available and used to order the reads. Such *reference-guided* assembly is still challenging due to the errors in the reads and because the reference often contains errors and gaps [41], [42]. Many of the assembly challenges are ameliorated if the target sequence is significantly oversampled and thus the information provided by short reads is highly redundant. This redundancy is quantified by means of a sequencing coverage – the average number of times a base in the target sequence is present in the

overlapping reads. However, the demands for higher throughput and lower sequencing costs often limit the coverage to medium (5-20X) or low ($\leq$ 5X). As an example, the ongoing *1000 Genomes Project* has opted for trading-off sequencing depth for the number of individuals being sequenced [46]. In its preliminary phase, the project has focused on sequencing a large number of individuals at a very low 3X coverage.

In reference-guided assembly, the short reads are first mapped to a reference sequence using an alignment algorithm (e.g., [6], [5]). Then each position along the target is determined by combining information provided by all the reads that cover that particular position. Due to the errors in base calls, short length of the reads, and repetitiveness in the target, both the mapping and the sequence assembly steps are potentially erroneous. The widely used tools to analyze and assemble genome sequence from high-throughput sequencing data include SAMtools [41] and Genome Analysis Toolkit (GATK) [42]. Note that both of these packages rely on the quality scores provided by the sequencing platform to infer the assembled sequence.

In this chapter, we formulate the reference-guided assembly problem as the inference of the genome sequence on a bipartite graph and solve it using a message-passing algorithm. Unlike existing state-of-the-art methods, the proposed algorithm performs reference-guided sequence assembly without relying on the, possibly inaccurate, quality scores of the short reads. Instead, it infers reliability of a base in the assembled sequence by combining the information of all the reads covering that particular position. The proposed algorithm can

be interpreted as the classical belief propagation under a certain prior. Binary reformulation of the problem leads to an alternative solution in the form of another message passing algorithm that is closely related to the power iteration method. The power iteration method approximates the solution to the sequence assembly problem by the leading singular vector of a matrix comprising read data. The power iteration method has guaranteed convergence, and its careful examination provides relation between the algorithm accuracy and the number of iterations. To evaluate the performance of proposed techniques, we compare them with a genie-aided maximum a posteriori (MAP) sequence assembly scheme which is an idealized assembler with perfect quality score information and error-free mapping of the reads to their locations. Results on both simulated and experimental data obtained by sequencing *Escherichia Coli* and *Neisseria Meningitidis* at UT Austin's Center for Genomic Sequencing and Analysis demonstrate that our proposed message-passing algorithm performs close to the aforementioned genie-aided MAP assembly scheme and is superior compared to state-of-the-art methods (in particular, it outperforms the aforementioned SAMtools and GATK software packages). Note that the developed algorithms as well as simulation and experimental studies are focused on haploid genomes – while modifications that enable application to diploid/polyploid genomes are relatively straightforward, they are beyond the scope of the current manuscript.

3.2 Contributions

In this chapter, we formulate the reference-guided sequence assembly problem as the inference of the genome sequence on a bipartite graph and solve it using a message-passing algorithm. The proposed algorithm can be interpreted as the well-known classical belief propagation scheme under a certain prior. Unlike existing state-of-the-art methods, the proposed algorithm combines the information provided by the reads without needing to know reliability of the short reads (so-called quality scores). Relation of the message-passing algorithm to a provably convergent power iteration scheme is discussed. To evaluate and benchmark the performance of the proposed technique, we find an analytical expression for the probability of error of a genie-aided maximum a posteriori (MAP) decision scheme. Results on both simulated and experimental data demonstrate that the proposed message-passing algorithm outperforms commonly used state-of-the-art tools, and it nearly achieves the performance of the aforementioned MAP decision scheme.

Implementation code of the ParticleCall algorithm in C++ is available at https://sourceforge.net/projects/mpsequencing/.

3.3 Graphical Model and the Message-Passing Assembly Algorithm

To facilitate processing of the short reads generated by next-generation sequencing instruments, we introduce a bipartite graph representing the reads and bases in the target sequence that needs to be assembled. The fundamental

building blocks of a sequence – the nucleotides A, C, G, and T – are numerically represented using 4-dimensional unit vectors containing a single non-zero component whose position indicates type of a nucleotide. In particular, the 4-dimensional unit vectors that we use are $\mathbf{e}_A = [1\ 0\ 0\ 0]^T$, $\mathbf{e}_C = [0\ 1\ 0\ 0]^T$, $\mathbf{e}_G = [0\ 0\ 1\ 0]^T$, and $\mathbf{e}_T = [0\ 0\ 0\ 1]^T$. Assume the target sequence has length L, and denote the bases in the sequence by $b_{1:L}$. Then each base in the target sequence is represented by a vector $\mathbf{b}_i \in \{\mathbf{e}_A, \mathbf{e}_C, \mathbf{e}_G, \mathbf{e}_T\}$. For convenience, we will assume that all the short reads at our disposal are generated by the same sequencing instrument and thus have identical read length l; note, however, that there is no loss of generality and that our scheme can combine reads generated by sequencing the same target on different instruments and of different read lengths. Let us denote the set of short reads by $\mathcal{R} = \{r_j\}$, $j = 1, 2, \ldots, n$. In general, the base calls in these reads are erroneous due to various uncertainties in the underlying sequencing-by-synthesis process. Average base-calling error rates of most current next-generation sequencing systems are on the order of 10^{-2}.

In reference guided sequence assembly, the short reads are mapped onto the reference using one among many recently developed short-sequence alignment algorithms [6], [5]. Note that the reads comprising bases with low quality scores are often discarded by the alignment algorithms. Ideally, the remaining reads (the ones of high fidelity) are accurately mapped to their corresponding locations on the reference sequence. However, for some reads there may exist several candidate positions which leads to possible mis-alignments and,

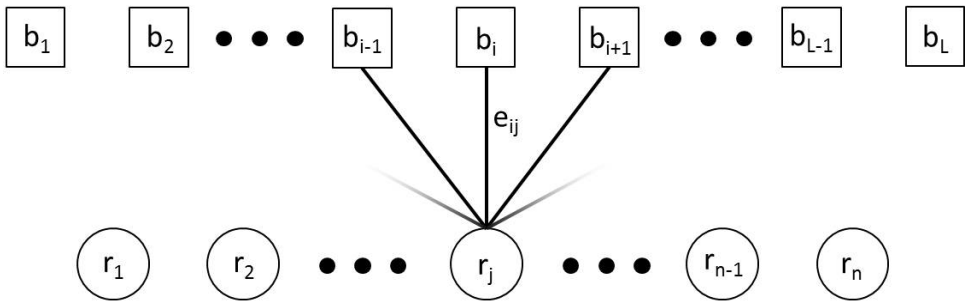

Figure 3.1: *Illustration of the reference-guided DNA sequence assembly problem using short reads. Nodes b_i represent bases in the target DNA sequence and r_j represent reads. Each read node is connected to l base nodes, where l denotes the read length.*

consequently, might provide erroneous information about the regions of the target sequence where the read is mis-aligned. As pointed out in [42], the misalignment rate for reads from genome regions which contain homozygous indels (where the chromosomes in a homologous pair have the same sequence but contain insertion or deletion as compared to the reference genome) can be as high as 15%.

We represent the reference-guided assembly problem by a graph shown in Fig. 3.1. The bipartite graph $G(b_{1:L} \cup r_{1:n}, E)$ illustrated in the figure has L base nodes (representing the target genome sequence) and n read nodes. Since we assume that all reads are of the same length l, each read node is connected to exactly l base nodes. The edge (i, j) in the edge set E connecting b_i and r_j is associated with a unit vector $\mathbf{e}_{ij}$ indicating information about the type of base b_i provided by the read r_j. Note that the bipartite graph described here is reminiscent of the graphical representation of the crowdsourcing problem in

[43]. Motivated by the iterative learning scheme proposed there, we employ a message-passing algorithm to infer the target genome sequence using overlapping reads. Note that the previously mentioned problem of having multiple candidate locations for mapping reads can be incorporated in the proposed graphical representation and resolved using the algorithm that we describe next.

The message passing algorithms rely on the exchange of messages between neighboring nodes in the graph [47]. Our algorithm operates on real-valued base messages $\{\mathbf{x}_{i\to j}\}_{(i,j)\in E}$ and read messages $\{y_{j\to i}\}_{(i,j)\in E}$. A base message $\mathbf{x}_{i\to j}$ is a 4×1 vector representing the likelihood of the base b_i being A, C, G, or T, while a read message $y_{j\to i}$ represents the reliability of read j. Read messages are initialized from a random distribution, and the message update rules at iteration k are given by

$$\mathbf{x}_{i\to j}^{(k)} \leftarrow \sum_{j'\in\partial i\setminus j} (2\mathbf{e}_{ij'} - \mathbb{1}) y_{j'\to i}^{(k-1)}, \tag{3.1}$$

$$y_{j\to i}^{(k)} \leftarrow \frac{1}{l-1} \sum_{i'\in\partial j\setminus i} \mathbf{e}_{i'j}^T \mathbf{x}_{i'\to j}^{(k)}, \tag{3.2}$$

where ∂i and ∂j denote collection of the neighboring nodes of nodes i and j, respectively, and $\mathbb{1}$ is a 4×1 vector containing all 1's. Note that $2\mathbf{e}_{ij'} - \mathbb{1}$ has element 1 in the position corresponding to the nucleotide base $b_{ij'}$ represented by $\mathbf{e}_{ij'}$, and -1's elsewhere. Hence a read with positive reliability value $y_{j'\to i}$ will increase the likelihood of $b_{ij'}$ and decrease the likelihood of other bases. Finally, the likelihood of a base being A, C, G, or T is calculated as the sum of the information provided by the reads weighted by each read's reliability.

The symbol with the highest likelihood is chosen as the estimate of the base in the corresponding position. The estimate rule for the i^{th} base is

$$\hat{b}_i = \arg \max_{t \in \{A,C,G,T\}} \mathbf{x}_i^{\{t\}}, \tag{3.3}$$

where the decision vector $\mathbf{x}_i = \sum_{j \in \partial i} (2\mathbf{e}_{ij} - \mathbb{1}) y_{j \to i}^{(k_m)}$. Here, k_m denotes the number of iterations performed and $\mathbf{x}_i^{\{t\}}$ denotes the likelihood corresponding to symbol $t \in \{A, C, G, T\}$ in the vector $\mathbf{x}_i = \left[\mathbf{x}_i^{\{A\}}\ \mathbf{x}_i^{\{C\}}\ \mathbf{x}_i^{\{G\}}\ \mathbf{x}_i^{\{T\}}\right]^T$. The procedure is formalized as Algorithm 3.

Note that Algorithm 3 needs to be appropriately initialized. In our experimental studies presented in Section 3.7, we initialize $y_{j \to i}^{(0)}$ by drawing from both Gaussian distribution $\mathcal{N}(1, 1)$ and uniform distribution $U[0, 1]$. For the data sets under consideration, it turns out that different initializations lead to identical solutions. The algorithm is terminated when the reliability increment between subsequent iterations is small, i.e., $\sum |y_{j \to i}^{(k)} - y_{j \to i}^{(k-1)}| < \epsilon$. As pointed out earlier, the algorithm does not require exact knowledge of quality scores, and iteratively infers reliability of individual reads.

Since the reads originating from a single sequencing instrument have identical lengths, the degree of the read nodes in the graph is uniform. On the other hand, degree of a base node is the number of reads that cover the corresponding base, usually referred to as the *sequencing coverage*. Typically, coverage varies from one position to another and, consequently, degree of the base nodes varies. Note that fragmentation of multiple copies of the target sequence – a fundamental step in shotgun sequencing procedure – can be viewed

Algorithm 3 Message passing for sequence assembly

Input: E, $\{\mathbf{e}_{ij}\}_{(i,j)\in E}$

1 Initialize read messages:

for all $(i,j) \in E$ **do**

Initialize $y_{j\to i}^{(0)}$;

end for

2 Iterations:

for $k = 1 \to k_m$ **do**

for all $(i,j) \in E$ **do**

Update base message:

$\mathbf{x}_{i\to j}^{(k)} \leftarrow \sum_{j'\in\partial i\setminus j}(2\mathbf{e}_{ij'} - \mathbb{1})y_{j'\to i}^{(k-1)}$;

Normalize $\mathbf{x}_{i\to j}^{(k)}$:

$\mathbf{x}_{i\to j}^{(k)} \leftarrow \frac{\mathbf{x}_{i\to j}^{(k)}}{||\mathbf{x}_{i\to j}^{(k)}||_2}$.

end for

for all $(i,j) \in E$ **do**

Update read message

$y_{j\to i}^{(k)} \leftarrow \frac{1}{l-1}\sum_{i'\in\partial j\setminus i}\mathbf{e}_{i'j}^T\mathbf{x}_{i'\to j}^{(k)}$;

end for

end for

3 Estimation:

for $i = 1 \to L$ **do**

Calculate decision vector $\mathbf{x}_i = \sum_{j\in\partial i}(2\mathbf{e}_{ij} - \mathbb{1})y_{j\to i}^{(k_m)}$;

Estimate the bases

$\hat{b_i} = \arg\max_{t\in\{A,C,G,T\}}\mathbf{x}_i^{\{t\}}$**;**

end for

as a uniform sampling from the original DNA strand. The resulting coverage is a random variable that can be described well by a Poisson distribution [49, 51].

Let $\bar{c}$ denote the average sequencing coverage. The computational complexity of the base message updating step (3.1), which needs to be performed in each iteration of Algorithm 3, is $\mathcal{O}(nl\bar{c})$ on average, while the complexity of the read message updating step (3.2) is $\mathcal{O}(nl^2)$. Since $L\bar{c} = nl$, the complexity of the algorithm is $\mathcal{O}(k_m nl(l+\bar{c})) = \mathcal{O}(k_m L\bar{c}(l+\bar{c}))$, where k_m denotes the number of iterations (i.e., the number of message updates). On the other hand, the simple plurality voting scheme has complexity $\mathcal{O}(L\bar{c}\log\bar{c})$. Our experimental studies show that $k_m \leq 30$ is sufficient for the convergence of the algorithm. We tested the algorithm on a broad range of parameters (in particular, for read lengths $l \leq 100$, coverage $\bar{c} \leq 60$), and found that the runtimes are comparable to those of the state-of-the-art techniques (SAMtools and GATK) – a specific comparison of runtimes is reported in Section 3.7.

3.4 Relation to standard belief propagation

As an alternative to the intuitively pleasing but basically heuristic message passing scheme proposed in Section 3.3, we can also derive a standard belief propagation algorithm for the reference-guided sequence assembly. To this end, we seek the sequence $\hat{b}_{1:L}$ that maximizes the joint probability $P(\hat{b}_{1:L}, p_{1:n})$, where $p_{1:n}$ denotes confidences of the aligned read data and

$p_j \in [0,1]$. This maximization can be formalized as

$$\max_{\hat{b}_{1:L}, p_{1:n}} \prod_{j=1}^{n} \mathcal{D}(p_j) \prod_{(i,j)\in E} \left\{ p_j \delta(\hat{b}_i = \mathbf{e}_{ij}) + \bar{p}_j \delta(\hat{b}_i \neq \mathbf{e}_{ij}) \right\}, \tag{3.4}$$

where $\mathcal{D}(p_i)$ denotes the prior distribution on p_i and $\bar{p}_j = 1 - p_j$. $\delta(\cdot)$ denotes an indicator function taking value 1 if its argument is true and is 0 otherwise. The joint optimization is computationally challenging and thus often practically not feasible. As an alternative, belief propagation provides an approximate solution to (3.4) by computing the marginal distributions of the optimization variables and selecting their most likely values according to the computed distributions. A thorough review of theoretical and practical aspects of the belief propagation method can be found in [45]. For the graphical model proposed in Section 3.3, we define two messages to facilitate belief propagation: $\tilde{x}_{i\to j}$ and $\tilde{y}_{j\to i}$. The former is the belief on $\hat{b}_i$ and essentially represents a distribution over the four possible nucleotide bases $\{\mathbf{e}_A, \mathbf{e}_C, \mathbf{e}_G, \mathbf{e}_T\}$. The latter is a probability of p_j on $[0,1]$. In the k^{th} iteration of the belief propagation algorithm, the message update rules are given by (see, e.g., [45] and the references therein)

$$\begin{aligned} \tilde{y}^{(k)}_{j\to i}(p_j) \;\propto\; & \mathcal{D}(p_j) \prod_{i'\in\partial j\setminus i} \sum_{m=A,C,G,T} \Big\{ p_j \delta(\mathbf{e}_{i'j} = \mathbf{e}_m) \\ & + \bar{p}_j \delta(\mathbf{e}_{i'j} \neq \mathbf{e}_m) \Big\} \tilde{x}^{(k)}_{i'\to j}(\mathbf{e}_m), \end{aligned} \tag{3.5}$$

$$\begin{aligned} \tilde{x}^{(k+1)}_{i\to j}(\hat{b}_i) \;\propto\; & \prod_{j'\in\partial i\setminus j} \int \Big(\tilde{y}^{(k)}_{j'\to i}(p_{j'}) (p_{j'} \delta(\hat{b}_i = \mathbf{e}_{ij'}) \\ & + \bar{p}_{j'} \delta(\hat{b}_i \neq \mathbf{e}_{ij'}) \Big) dp_{j'}. \end{aligned} \tag{3.6}$$

After the completion of the iterative procedure, the bases b_i in the target genome are estimated by first computing the beliefs $\tilde{x}_i(\hat{b}_i) \propto$

$$\prod_{j' \in \partial i} \int \Big(\tilde{y}^{(k)}_{j' \to i}(p_{j'}) (p_{j'} \delta(\hat{b}_i = \mathbf{e}_{ij'}) + \bar{p}_{j'} \delta(\hat{b}_i \neq \mathbf{e}_{ij'})) \Big) dp_{j'}, \tag{3.7}$$

where $\hat{b}_i \in \{\mathbf{e}_A, \mathbf{e}_C, \mathbf{e}_G, \mathbf{e}_T\}$, and then choosing the base with the highest $\tilde{x}_i$ value. Note that, by exploiting the symmetry of the expression (3.4), we can write

$$\begin{aligned} \tilde{x}^{(k+1)}_{i \to j}(\hat{b}_i \neq \mathbf{e}_m) \quad \propto \quad & \prod_{j' \in \partial i \setminus j} \int \Big(\tilde{y}^{(k)}_{j' \to i}(p_{j'}) (p_{j'} \delta(\mathbf{e}_{ij'} \neq \mathbf{e}_m) \\ & + \bar{p}_{j'} \delta(\mathbf{e}_{ij'} = \mathbf{e}_m) \Big) dp_{j'}. \end{aligned}$$

For the brevity of notation, we denote $\tilde{x}^{(k)}_{i \to j}(\mathbf{e}_m) = \tilde{x}^{(k)}_{i \to j}(\hat{b}_i = \mathbf{e}_m)$. Assuming that the prior distribution on p_j, $\mathcal{D}(p_j)$, is Beta(0,0) (which is essentially as same as the Bernoulli(1/2) distribution), the read confidence is a binary variable,

$$p_j = \begin{cases} 0, \text{ w.p. } 1/2, \\ 1, \text{ w.p. } 1/2. \end{cases}$$

Define a log-likelihood ratio

$$y^k_{j \to i} = \log \Big(\frac{\tilde{y}^{(k)}_{j \to i}(1)}{\tilde{y}^{(k)}_{j \to i}(0)} \Big). \tag{3.8}$$

After substituting (3.5) in (3.8), we obtain

$$\begin{aligned} y^{(k)}_{j \to i} \quad &= \quad \sum_{i' \in \partial j \setminus i} \log \frac{\tilde{x}^{(k)}_{i' \to j}(\mathbf{e}_{i'j})}{\sum_{\mathbf{e}_m \neq \mathbf{e}_{i'j}} \tilde{x}^{(k)}_{i' \to j}(\mathbf{e}_m)} \\ &= \quad \sum_{i' \in \partial j \setminus i} \log \frac{\tilde{x}^{(k)}_{i' \to j}(\mathbf{e}_{i'j})}{\tilde{x}^{(k)}_{i' \to j}(\hat{b}_{i'} \neq \mathbf{e}_{i'j})}. \end{aligned} \tag{3.9}$$

$$
\begin{aligned}
\mathbf{x}_{i\to j}^{(k)}(1) &= \log \frac{x_{i\to j}^{(k)}(\mathbf{e}_A)}{x_{i\to j}^{(k)}(\hat{b}_i \neq \mathbf{e}_A)} \\
&= \sum_{j'\in\partial i\backslash j} \log \frac{\int \left(\tilde{y}_{j'\to i}^{(k-1)}(p_{j'})(p_{j'}\delta(\mathbf{e}_{ij'}=\mathbf{e}_A)+\bar{p}_{j'}\delta(\mathbf{e}_{ij'}\neq\mathbf{e}_A)\right) dp_{j'}}{\int \left(\tilde{y}_{j'\to i}^{(k-1)}(p_{j'})(p_{j'}\delta(\mathbf{e}_{ij'}\neq\mathbf{e}_A)+\bar{p}_{j'}\delta(\mathbf{e}_{ij'}=\mathbf{e}_A)\right) dp_{j'}} \\
&= \begin{cases} \log \frac{\tilde{y}_{j'\to i}^{(k-1)}(1)}{\tilde{y}_{j'\to i}^{(k)}(0)} = y_{j'\to i}^{(k-1)} & \text{if } \mathbf{e}_{ij'}=\mathbf{e}_A \\ -\log \frac{\tilde{y}_{j'\to i}^{(k-1)}(1)}{\tilde{y}_{j'\to i}^{(k)}(0)} = -y_{j'\to i}^{(k-1)} & \text{if } \mathbf{e}_{ij'}\neq\mathbf{e}_A \end{cases}
\end{aligned} \tag{3.11}
$$

Define a 4×1 vector message $\mathbf{x}_{i\to j}^{(k)}$ as

$$
\mathbf{x}_{i\to j}^{(k)} = \left[\mathbf{x}_{i\to j}^{(k)}(1) \quad \mathbf{x}_{i\to j}^{(k)}(2) \quad \mathbf{x}_{i\to j}^{(k)}(3) \quad \mathbf{x}_{i\to j}^{(k)}(4)\right]^T,
$$

where

$$
\begin{aligned}
\mathbf{x}_{i\to j}^{(k)}(1) &= \log \frac{\tilde{x}_{i\to j}^{(k)}(\mathbf{e}_A)}{\tilde{x}_{i\to j}^{(k)}(\hat{b}_i\neq\mathbf{e}_A)}, \\
\mathbf{x}_{i\to j}^{(k)}(2) &= \log \frac{\tilde{x}_{i\to j}^{(k)}(\mathbf{e}_C)}{\tilde{x}_{i\to j}^{(k)}(\hat{b}_i\neq\mathbf{e}_C)}, \\
\mathbf{x}_{i\to j}^{(k)}(3) &= \log \frac{\tilde{x}_{i\to j}^{(k)}(\mathbf{e}_G)}{\tilde{x}_{i'\to j}^{(k)}(\hat{b}_{i'}\neq\mathbf{e}_G)}, \\
\mathbf{x}_{i\to j}^{(k)}(4) &= \log \frac{\tilde{x}_{i\to j}^{(k)}(\mathbf{e}_T)}{\tilde{x}_{i'\to j}^{(k)}(\hat{b}_{i'}\neq\mathbf{e}_T)}.
\end{aligned}
$$

It is straightforward to write

$$
y_{j\to i}^{(k)} = \sum_{i'\in\partial j\backslash i} \mathbf{e}_{i'j}\mathbf{x}_{i'\to j}^{(k)}. \tag{3.10}
$$

A closer examination of the first element of $\mathbf{x}_{i\to j}^{(k)}$, $\mathbf{x}_{i\to j}^{(k)}(1)$, leads to the simplification shown in (3.11), where we implicitly used the assumption that p_j is binary. We can obtain similar expressions to (3.11) for other components of

$\mathbf{x}_{i\to j}^{(k)}$. As a result, the updating rule for $\mathbf{x}_{i\to j}^{(k)}$ simplifies to

$$\mathbf{x}_{i\to j}^{(k)} = \sum_{j'\in\partial i\setminus j} (2\mathbf{e}_{ij'} - \mathbb{1}) y_{j'\to i}^{(k-1)}, \tag{3.12}$$

where the vector $2\mathbf{e}_{ij'} - \mathbb{1}$ has element 1 in the position corresponding to the nucleotide base $b_{ij'}$ represented by $\mathbf{e}_{ij'}$, and -1's elsewhere. Therefore, the belief propagation update rule (3.12) is identical to the update rule (3.1) of our message passing algorithm presented in Section 3.3. Moreover, the update rule (3.10) is identical (up to the scaling factor) to the message update rule (3.2). Therefore, the message passing scheme proposed in Section 3.3 can be interpreted as belief propagation under a specific prior on the confidence of the aligned data p_j – in particular, p_j should come from a Beta(0,0) distribution, i.e., be treated as a binary variable.

3.5 Binary representation, message passing, and power iteration algorithm

So far, we discussed reference-guided assembly schemes that rely on a representation of the nucleotide basis with 4-dimensional vectors $\{\mathbf{e}_A, \mathbf{e}_C, \mathbf{e}_G, \mathbf{e}_T\}$. As an alternative, in this section we rely on a binary representation of nucleotides to formulate a message passing scheme and discuss the provably convergent power iteration algorithm for finding the target genome sequence. The power iteration scheme finds the desired sequence by computing the leading singular vectors of an appropriately defined data matrix.

The four-letter alphabet $\{A, C, G, T\}$ in DNA sequencing data can be

represented using binary symbols, e.g., $\{+1, -1\}$. In particular, we encode the nucleotide basis as $A = \{-1, -1\}$, $C = \{-1, +1\}$, $G = \{+1, -1\}$, and $T = \{+1, +1\}$, and represent reads as binary sequences comprising $\{\pm 1\}$. Similar to how we built a model utilizing 4-dimensional vectors $\{\mathbf{e}_A, \mathbf{e}_C, \mathbf{e}_G, \mathbf{e}_T\}$ in Section 3.3, we define a bipartite graph where each base b_i is represented by two binary nodes $\tilde{b}_{2i-1}$ and $\tilde{b}_{2i}$. Using the output of an alignment algorithm, each read node of the bipartite graph is connected to $2l$ binary base nodes in the node set $\tilde{b}_{1:2L}$, where l denotes read length and L is the length of the target sequence. For convenience, let us denote the resulting graph by $G(\tilde{b}_{1:2L} \cup r_{1:n}, \tilde{E})$. The edge (k, j) in $\tilde{E}$ connecting $\tilde{b}_k$ and r_j is assigned a variable $e_{kj} \in \{\pm 1\}$, the binary representation of $\tilde{b}_k$ provided by read r_j. Given such a graphical representation, we can apply a binary message passing algorithm as in [43]. In particular, the read and base messages are scalars and the update equations are given by

$$x_{i\to j}^{(k)} \leftarrow \sum_{j' \in \partial i \backslash j} e_{ij'} y_{j'\to i}^{(k-1)}, \tag{3.13}$$

$$y_{j\to i}^{(k)} \leftarrow \sum_{i' \in \partial j \backslash i} e_{i'j}^{T} x_{i'\to j}^{(k)}. \tag{3.14}$$

After the iterative procedure reaches a stopping criterion, the binary string representing unknown target DNA sequence is obtained as the weighted average

$$\tilde{b}_i = \text{sign}\Big(\sum_{j' \in \partial i \backslash j} e_{ij'} y_{j'\to i}\Big). \tag{3.15}$$

The above algorithm is known to converge to the optimal solution when the bi-partite graph is regular [44]. In our application, however, the graph is not

regular since the sequencing coverage varies. Nevertheless, we find that the binary message-passing algorithm performs very well in both simulations and on experimental data, as we demonstrate in Section 3.7. The binary message passing algorithm is also closely related to the power iteration scheme for computing the leading singular vector of an appropriately defined data matrix. We next examine the power iteration algorithm and argue its convergence.

With the adopted binary encoding of nucleotides, we can represent sequencing reads by a sparse $n \times 2L$ matrix D. The $2L$ columns of D correspond to the L positions in the target sequence whereas the j^{th} row of D comprises binary data representing read r_j. In each row, only $2l$ entries are non-zero (representing an l-long read) while the remaining ones are filled with zeros. Therefore, matrix D has entries $D_{ij} \in \{0, +1, -1\}$. Since the percentage of nonzero entries of D is $\frac{2l}{L}$ and $L \gg l$, D is a sparse matrix. It is easy to show (see, e.g., [44]) that if each row of D has the same number of nonzero entries, and the same holds for each column, the left singular vector corresponding to the largest singular value of D is a reliable estimate of the target genome sequence when the measurement noise (i.e., read error rate) is low. Here is an illustration. Let s denote the $2L \times 1$ binary vector with alphabet $\{-1, +1\}$ representing the true sequence of length L, and let the number of nonzero entries in each columns of D be c. Consider the case where the reads are error-free and s is a $2L \times 1$ all one vector $\mathbb{1}_{2L}$. Since $DD^T\mathbb{1}_{2L} = 2Lc\mathbb{1}_{2L}$, then s is an eigenvector of DD^T. Here D is a non-negative matrix with entries 0s and 1s and thus, by Perron-Frobenius theorem, $\mathbb{1}_{2L}$ is a left singular vector

corresponding to D's largest singular value. In the general case where s consists of both 1 and -1, we can represent $s = S\mathbb{1}_{2L}$ where S is a $2L \times 2L$ diagonal matrix with $\text{diag}(S) = s$. In this case, it is straightforward to generalize the above analysis and show that s remains to be proportional to the leading singular vector of the matrix D.

Performing singular value decomposition is roughly cubic in the dimension of D and, for our problem dimensions, clearly infeasible. Fortunately, we only need to find $\mathbf{u}$, the leading singular vector of D, and then estimate the target sequence s as $\text{sign}(\mathbf{u})$. This can be done in a computationally efficient way using the power iteration technique due to sparsity of D. In particular, the power iteration procedure entails computing

$$\mathbf{x}^{(k)} = D\mathbf{y}^{(k-1)}, \quad \mathbf{y}^{(k)} = D^T\mathbf{x}^{(k)}. \tag{3.16}$$

To demonstrate convergence of the power iteration scheme (3.16), let us denote the singular values of D as $\sigma_i(D)$, where $\sigma_1(D) \geq \sigma_2(D) \geq ... \geq 0$. With a random initialization $\mathbf{y}^{(0)}$, power iterations will converge to the singular vector $\mathbf{u}$ if the inequality $\sigma_1(D) > \sigma_2(D)$ holds strictly. The speed of the convergence of power iterations depends on the ratio $\sigma_2(D)/\sigma_1(D)$. This can be easily shown by an analysis of the consecutive projections of the iteratively updated vectors $\mathbf{x}^{(k)}$ onto the singular vector $\mathbf{u}$. In particular, the projection of $\mathbf{x}^{(k)}$ onto $\mathbf{u}$ is $(\mathbf{u}^T\mathbf{x}^{(k)})\mathbf{u}$. A closer look into the singular value decomposition shows that $\mathbf{u}^T\mathbf{x}^{(k)}\mathbf{u} = (\sigma_1(D))^2\mathbf{u}^T\mathbf{x}^{(k-1)}\mathbf{u}$ and $(\mathbf{x}^{(k)} - \mathbf{u}^T\mathbf{x}^{(k)}\mathbf{u}) \leq (\sigma_2(D))^2(\mathbf{x}^{(k-1)} -$

$\mathbf{u}^T\mathbf{x}^{(k-1)}\mathbf{u}$). Therefore,

$$\begin{aligned}\frac{||\mathbf{x}^{(k)} - \mathbf{u}^T\mathbf{x}^{(k)}\mathbf{u}||}{||\mathbf{u}^T\mathbf{x}^{(k)}\mathbf{u}||} &\leq \left(\frac{\sigma_2(D)}{\sigma_1(D)}\right)^2 \frac{||\mathbf{x}^{(k-1)} - \mathbf{u}^T\mathbf{x}^{(k-1)}\mathbf{u}||}{||\mathbf{u}^T\mathbf{x}^{(k-1)}\mathbf{u}||} \\ &\leq \left(\frac{\sigma_2(D)}{\sigma_1(D)}\right)^{2k} \frac{||\mathbf{x}^{(0)} - \mathbf{u}^T\mathbf{x}^{(0)}\mathbf{u}||}{||\mathbf{u}^T\mathbf{x}^{(0)}\mathbf{u}||}.\end{aligned}$$

Clearly, power iterations will converge with any initialization if $\sigma_1(D) > \sigma_2(D)$, and the speed of convergence depends on the ratio of $\sigma_1(D)$ and $\sigma_2(D)$ – the larger the ratio, the faster the convergence. On the other hand, from (3.16) it directly follows that the update equations for the entries of $\mathbf{x}^{(k)}$ and $\mathbf{y}^{(k)}$ can be written as

$$x_i^{(k)} = \sum_{j\in\partial i} D_{ij} y_j^{(k-1)}, \quad y_j^{(k)} = \sum_{i\in\partial j} D_{ij} x_i^{(k)}. \tag{3.17}$$

Note that the power iterations (3.17) differ from the message update rules (3.13) and (3.14) in only one term. As our results in Section VI show, accuracy of message passing and power iterations is essentially identical, while the former converges in significantly fewer iterations than the latter. Moreover, both message-passing schemes – the one based on the representation of basis via 4-dimensional vectors $\{\mathbf{e}_A, \mathbf{e}_C, \mathbf{e}_G, \mathbf{e}_T\}$ as well as the one relying on the binary representation of nucleotides – converge after approximately the same number of iterations.

3.6 Benchmarking performance of the proposed assembly scheme

To evaluate performance of the proposed iterative learning scheme, in this section we compare the probability of error with a genie-aided maximum

a posteriori (MAP) estimator of the bases in the target sequence. The genie-aided MAP estimator considers an idealized scenario where short reads are mapped to the reference genome with no errors, i.e., there are no misplacements of the reads along the reference sequence. Moreover, it assumes that the exact probabilities of mis-calling the bases in the short reads are available (i.e., has exact quality score information). Recall that our message-passing scheme does not make such practically unrealistic assumptions and, in fact, does not require prior knowledge of quality scores. The details of the MAP estimator can be found in [37]. For the sake of completeness, we review the basic computing procedure here.

Let b_k denote the k^{th} base in the target sequence, and let $y_k^{(i)}$ denote the signal generated by sequencing b_k, $i = 1, 2, \ldots, c_k$, where c_k denotes the total number of reads covering b_k. Assume that the probability of erroneously calling b_k in the i^{th} read is $p_k^{(i)}$. Given the base calls of the reads covering b_k, $y_k^{(i)}$, the MAP estimate $\hat{b}_k$ is found as

$$\hat{b}_k = \arg\max_x \sum_{i=1}^{c_k} \delta(y_k^{(i)} = x) w_k^{(i)} + \log(p_k^{(i)}) + \log(P(b_k = x)),$$

where $\delta(\cdot)$ denotes an indicator function taking value 1 if its argument is true and is 0 otherwise, and we introduced $w_k^{(i)} = \log\left(\frac{1-p_k^{(i)}}{p_k^{(i)}}\right)$. Therefore, the estimate $\hat{b}_k$ is given by

$$\hat{b}_k = \arg\max_x \sum_{i=1}^{c_k} w_k^{(i)} \delta(y_k^{(i)} = x) + \log(P(b_k = x)). \tag{3.18}$$

In the absence of prior information $P(b_k = x)$, the MAP estimation of b_k in (3.18) is identical to the so-called weighted plurality voting [48].

3.7 Results

In this section we present performance studies using both simulations and experimental data sets. First, using realistic synthetic data, we compare the performance of the message passing algorithm from Section 3.3 (Algorithm 3), the binary message passing algorithm and the power iteration algorithm. Moreover, we examine the convergence properties of all these schemes and benchmark their accuracy by comparing it with the genie-aided MAP estimation employed in the idealistic scenario where the exact error probabilities of the reads are known. Then we proceed by testing the algorithms on the experimental data we obtained by sequencing *E. Coli* and *N. Meningitides* using Illumina's HiSeq sequencing instrument that provides 100-bp long reads. In particular, we compare the performance of our developed reference-guided sequence assembly algorithms with the commonly used sequencing data analysis tools including GATK and SAMtools.

3.7.1 Simulation data

We simulated reference-guided sequence assembly of the genome of a strain of *Neisseria Meningitidis*. The reference sequence is obtained from the GenBank (http://www.ncbi.nlm.nih.gov/nuccore) database and is $L = 2,184,406$ bases long. The reference is used to generate target sequences having 1% variation rate. We then uniformly select starting positions along the sequence and simulate short reads of length $l = 76$ (mimicking Illumina's Genome Analyzer II platform). Sequencing errors in these reads are simulated

according to the position-dependent base calling error profile typical of this particular sequencing platform [12]. The average error rate of the base calling procedure is 0.015 (averaged over all reads and bases in the reads). To construct the bipartite graphical model, we map the reads to the reference sequence using an alignment algorithm based on the Burrows-Wheeler transform [6] and thus establish connections (i.e., edges) between the read nodes and their aligned base nodes. The read nodes with multiple candidate mapping positions are replicated (where each replica may be assigned different confidence score), and each replica is connected to its corresponding set of base nodes. The bipartite graph with binary base nodes introduced in Section 3.5 is constructed in the same way. We apply both the message passing algorithms from Section 3.3 and Section 3.5 to infer the target sequence (note that since the algorithms are randomly initialized, the stopping points and hence the resulting assembled sequences may be different). We also form the binary data matrix representing all the short read data and employ the power iteration method to infer the target genome sequence. While the analysis in Section 3.5 gives a guarantee of convergence of the power iteration algorithm, we found that its convergence is usually faster than the theoretical bound. We set the stopping criterion for all these iterative learning methods as $\sum |y_{j\to i}^{(k)} - y_{j\to i}^{(k-1)}| < \epsilon = 0.01L$. It turns out that both message passing algorithms need ~ 30 iterations to converge, while the power iterations converge in ~ 50 iterations. We initialize all these algorithm by generating $y_{j\to i}^{(0)}$ from the Gaussian distribution $\mathcal{N}(1,1)$ and the uniform distribution $U(0,1)$ – our extensive simulation studies indicate that

different initializations lead to the same error rate of the considered iterative schemes.

For a comparison, we also consider the plurality voting based decision scheme often used in practice (see, e.g., [49]). Here, multiple calls for a base in any given position along the target sequence are consolidated by performing plurality voting. Notice that, in both message passing and plurality voting, we assume the error profiles of the reads (i.e., base calling error rates) are unknown. Plurality voting assumes all reads have equal reliability while message passing scheme iteratively infers the reliability of each read. We also consider probability of error of the MAP decision scheme in Section 3.6 which assumes perfect knowledge of the positions of reads along the target sequence and exact information about position-dependent base calling errors (both assumptions are unrealistic in practice). The error rates of these algorithms are shown in Fig. 3.2 for various sequencing coverages (horizontal axis shows the average coverage). As can be seen from Fig. 3.2, the message-passing scheme and the power iteration algorithm outperform plurality voting. The binary message passing algorithm has almost identical accuracy as power iterations, while being slightly worse than Algorithm 3. Moreover, we see that the error rates of message passing are close to the genie-aided MAP decision scheme, which represents the best that can be achieved.

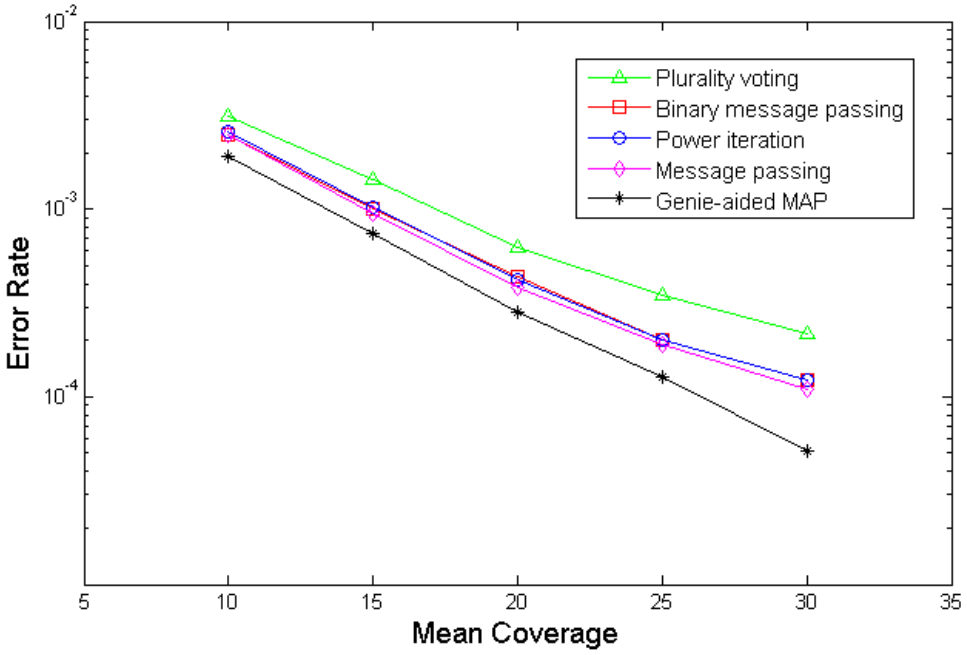

Figure 3.2: *Error rates performance of the iterative learning schemes (message passing, binary message passing, and power iterations) compared with the plurality voting and genie-aided MAP schemes. The error rates of iterative learning schemes and plurality voting are averaged over* 20 *experiments. Note that, as seen in the figure, power iteration and binary message passing have almost identical performance.*

3.7.2 Experimental data

In addition to the simulation studies, we tested the performance of our proposed iterative learning schemes for reference-guided sequence assembly using two experimental data sets. In particular, we sequenced *Escherichia Coli* (from strain MG1655, 4.64×10^6 bases long) and *Neisseria Meningitidis* (from strain FAM18, having length 2.2×10^6) at the Center for Genomic Sequencing and Analysis of the University of Texas at Austin. The data is obtained using Illumina's HiSeq platform that provides 100bp-long paired-end reads, and the performance of our proposed methods are compared with that of the widely used sequencing analysis packages SAMtools and GATK. Both SAMtools and GATK process aligned next-generation sequencing data stored in SAM format, the alignment file format provided by the majority of frequently used alignment tools (e.g., BWA). These files contain the aligned reads, their positions and the quality scores of the bases. SAMtools calculates empirical quality scores from the alignment information and uses them to recalibrate the raw quality scores provided by the sequencing platform. The assembled sequence is formed using the aligned bases weighted by these new quality scores. In addition to the quality score recalibration, GATK also performs a local realignment procedure to correct misaligned reads, especially from the target genome region containing indels compared to the reference genome. After performing sequence assembly using quality score information, these software packages can also perform downstream single nucleotide polymorphism (SNP) detection, while GATK also incorporates a machine learning tool to separate

true variation from sequencing platform artifacts.

The two genomes are sequenced using 67% of an HiSeq platform lane having approximately 30×10^6 reads, resulting in the coverage greater than 200. This enables accurate inference of the true *E. Coli* and *N. Meningitides* sequences using any of the techniques discussed in the chapter, providing us with the ground truth. To determine the accuracy of our proposed schemes in realistic scenarios where the coverage is limited, we uniformly subsample the data to emulate low coverage situations. The resulting error rates are shown in Table 3.7.2. As can be seen there, the developed message passing schemes outperform both SAMtools and GATK in terms of the accuracy. The number of iterations for each message passing scheme was set to 30, which at coverage $c = 20$ resulted in the average CPU runtimes of 65 and 37 minutes for processing *E. Coli* and *N. Meningitidis* data sets, respectively (the algorithms were coded in C++, run on a 3.07G Hz single core machine). The corresponding runtimes for SAMtools are 50 and 28 minutes, and for GATK 53 and 30 minutes. As seen from the table, increasing the coverage can dramatically improve accuracy of the assembly – recall the discussion from Section 3.6 where we showed that the probability of error of the genie-aided MAP estimator decreases exponentially with the coverage. However, increasing coverage also increases the cost of the sequencing project.

Note that the sequenced genome might contain insertions as compared to the reference or, equivalently, the reference sequence contains gaps. This structural variation can be detected in the alignment stage by using paired-

Sequence and Coverage	Number of errors			
	MP	BMP	SAMtools	GATK
E coli				
15	3484 ± 88	3507 ± 76	3655 ± 66	3598 ± 72
20	2566 ± 66	2599 ± 54	2677 ± 71	2634 ± 53
25	1243 ± 31	1256 ± 44	1298 ± 41	1283 ± 55
30	763 ± 20	781 ± 23	811 ± 23	798 ± 18
N. Meningitidis				
15	2168 ± 35	2231 ± 43	2404 ± 37	2358 ± 30
20	1201 ± 29	1299 ± 26	1388 ± 30	1379 ± 20
25	899 ± 16	913 ± 20	933 ± 24	921 ± 19
30	658 ± 11	669 ± 11	681 ± 9	680 ± 15

Table 3.1: Performance of the message passing algorithm (MP), binary message passing algorithm (BMP), SAMtools and GATK on E. coli and N. Meningitidis sequencing data with various coverages. The average number of decision errors and the corresponding standard deviation are computed over 30 runs.

end reads [52], [53]. The paired-end reads have a known range of lengths of inserts between the reads in a pair. The gaps in the reference can be detected by relying on a multi-read alignment of the pairs of reads and comparing the aligned positions with the insert lengths. We used the scheme in [53] to perform the alignment of our *E. Coli* data set and detected 34 gaps in the reference. We included the gap positions as additional base nodes in our graphical model and used our Algorithm 3 to identify the order of nucleotides in the gaps. As a result, 31 out of 34 gaps were reconstructed (i.e., closed).

3.8 Conclusions

We studied reference-guided sequence assembly from short reads generated by next-generation sequencing technologies, specifically focusing on the problem of obtaining the target genome sequence from potentially erroneous and misaligned reads. We cast the problem as the inference of the target sequence on an appropriately defined bipartite graph and proposed iterative learning algorithms for solving it. In particular, we developed message passing algorithms that rely on both binary as well as representation of nucleotide bases by 4-dimensional vectors. It was shown that the derived message passing algorithm (in particular, Algorithm 3 in Section 3.3) can be interpreted as the standard belief propagation under a certain prior. In addition, the problem was rephrased so that the power iteration algorithm, employed to find the leading singular vector of a matrix collecting all short reads, results in a good approximation of the target sequence. Convergence of power iterations is guaranteed, while the convergence of message passing algorithms is studied empirically. Unlike existing methods, the proposed algorithms find the desired sequence without using reliability information (i.e., quality scores) of the short reads – in fact, message passing algorithms infer the aforementioned quality score information.

To assess achievable accuracy of the proposed iterative learning techniques, we analyzed the probability of error of a genie-aided maximum a posteriori decision scheme in the idealized scenario where the base calling error rates and read mapping locations are known perfectly. It was shown empiri-

cally that the iterative learning schemes perform close to the genie-aided estimation scheme, and that they outperform state-of-the-art software packages for downstream processing of sequencing data.

Chapter 4

Error correction in de novo sequence assembly using quality information

4.1 Background

In Chapter 3, we considered the sequence reconstruction problem under the assumption we have a reference sequence of the same species. In *de novo* sequence assembly, however, we need to assemble the long target genome sequence directly from short reads without a reference. Several recent developed *de novo* assembly algorithms can be found in [7–10]. These algorithms are highly sensitive to read errors. As we see in Chapter 3, the per-base read error rates of next-generation sequencing are between 1% to 2%, which is much higher compared to traditional Sanger technique. To improve the accuracy of the assembly using next-generation sequencing data, a key step is to correct the errors in the raw short reads using the redundancy information provided by the high coverage of next-generation sequencing. A set of short read error correction methods optimized for next-generation sequencing have been introduced to tackle this task. SHREC [54] uses a generalized suffix tree data structure to improve the efficiency of error correction. The recently developed HiTEC [55] utilizes a suffix array structure built on the string of all reads. The parameters of the algorithm are optimized using statistical analysis of the

correction procedure. This algorithm is shown to be more accurate and robust than SHREC and other previous methods. However, it assumes the reads provided by the next-generation sequencing have flat per-base error rate, which limits the accuracy of this method.

In this chapter, we develop a new short read error correction method for *de novo* assembly of next-generation sequencing. Traditional error correction methods are based on the assumption that all bases in the reads have same probability of error, which is not realistic to next-generation sequencing data. Our method considers realistic error profile of the reads in next-generation sequencing technology and utilizes the provided base quality information to improve the accuracy of error correction. A hypothesis testing scheme relying on quality scores is used to improve the overlap detection. By using the suffix array data structure, it can also efficiently search all potential support segments in the reads. Experimental results show that our algorithm can improve the error correction performance comparing with the previous mentioned methods.

4.2 Suffix arrays and its application to sequencing data

Suffix array is a simple data structure first introduced to perform on-line string search by Manber and Myers [56]. Compared to suffix trees, suffix arrays are more space-efficient in applications while their complexity of searching substrings is competitive. A DNA sequence can be represented as a string with an appropriately defined alphabet and the corresponding suffix array can

be obtained. Here we give a brief overview of suffix array data structure and its application to DNA sequences.

Consider the 4-letter alphabet representing the nucleotide bases $\Sigma = \{A, C, G, T\}$; then a DNA sequence is a string over Σ, and we denote the set of all DNA sequences by Σ^*. Assume that the length of a DNA sequence S is l. The suffix of S starting at the i-th position is denoted by $S_i = s_i s_{i+1} \dots s_l$. The reverse compliment of S, denoted by $\bar{S}$, is a sequence obtained by reversing S and then complementing its bases, i.e., $A \leftrightarrows T$, $C \leftrightarrows G$.

The basis of the suffix array data structure is an alphabetically sorted array of all the suffixes of string S. We denote suffix array as SA. $SA[k]$ is the start position of the kth smallest suffix in the suffix set $\{S_1, S_2, ...S_l\}$. Therefore, we can order the suffixes as $S_{SA[1]} < S_{SA[2]} < \ldots < S_{SA[l]}$. The longest common prefix between consecutive suffixes in the suffix array is denoted by lcp. $lcp[i]$ is the length of the longest common prefix between $S_{SA[i-1]}$ and $S_{SA[i]}$. $lcp[0] = 0$ by definition. Suffix arrays and the corresponding lcp can be computed efficiently in $O(l)$ time and space using the algorithms in [57, 58] and [59].

We here apply the concept of suffix arrays to the DNA sequencing data sets in the *de novo* assembly problem. In high-throughput DNA sequencing, a target genome with length L is oversampled and a large set of short reads with same length are generated. These reads are potentially erroneous as discussed in Chapter 2 and Chapter 3. Assume there are n reads $r_1, r_2, \dots, r_n$ with same length l. In next-generation sequencing, we also have the quality scores of all

these reads which indicate their probability of error. Note that this is a major difference between our approach and existing methods where the quality scores are not considered and the error probability is assumed to be the same for all bases.

We concatenate all the reads r_is and their complements $\bar{r}_i$s to construct a string and use \$ as an auxiliary symbol to mark the end of each read. The resulting string is

$$R = r_1\$\bar{r}_1\$r_2\$\bar{r}_2\$ \ldots r_n\$\bar{r}_n\$.$$

We obtain suffix array SA and longest common prefix lcp of read string R. Our basic error correction mechanism follows the idea described in both HiTEC and SHREC where the support values of a base is counted. Assume an erroneous base is in the kth position of read r_i, which is sampled from the jth position of the genome G. In order to correct the error, we look for bases in other reads having the same *support*. The support u of a base b is defined as a fixed length segment of bases in the reads preceding b. We define the number of occurrences of pair (u, b) in all reads as $\mathbf{supp}(u, b)$. Intuitively, the larger $\mathbf{supp}(u, b)$ is, the more confident we are in the correctness of base b. If we observe another base c which has the same support u but $\mathbf{supp}(u, c)$ is small, we are likely to identify c as an erroneous called base in the reads and should be corrected into b.

4.3 Optimization of the parameters using quality scores

There are two parameters that need to be calculated to run the error correction algorithm: the length of the support w and the threshold T for verification of the correctness of a support sequence. By exploiting the error profile of the sequencing data sets, we can obtain the optimal values of these parameters under some criterion.

4.3.1 Calculate optimal threshold

The length of the support u affects the performance of error correction. Based on the quality scores provided by the sequencing platforms, we can find the optimal support length to minimize the expected number of errors. Assume the support length is w, and that a base b inside the read has support sequence u. We want to calculate the expected number of correct (u, b) pairs and erroneous pairs. We assume that the error profile (average probability of error for different positions) of these short read data is known and can be transformed to the probability of being correct $q(1), q(2), \ldots, q(l)$ for each read position.

We denote p_c as the probability of a read covers (u, b) pair and contains no errors inside this pair. We also assume the read is sampled uniformly from the original genome G. If the (u, b) pair starts from jth position of the genome G, the read covering this pair can start from $j-l+w+1, j-l+w+2, \ldots, j$th

positions. So we can get

$$\begin{aligned} p_c &= \frac{l-w}{L}\frac{1}{l-w}\{q(1)q(2)\dots q(w+1)+q(2)q(3)\dots q(w+2) \\ &+ \dots + q(l-w)\dots q(l-1)q(l)\} \\ &= \frac{1}{L}\sum_{i=1}^{l-w} q(i)q(i+1)\dots q(i+w) \end{aligned} \tag{4.1}$$

Let N_c denote the number of such reads, the probability of $N_c = k$ is

$$P(N_c = k) = \binom{n}{k} p_c^k (1-p_c)^{n-k}.$$

In genome G, there are in total $L - w \approx L$ (u, b) pairs. Thus the expected number of correct (u, b) pairs with $\mathbf{supp}(u, b) = k$ is

$$\begin{aligned} M_c(k) &= P(N_c = k)L \\ &= \binom{n}{k} p_c^k (1-p_c)^{n-k} L. \end{aligned} \tag{4.2}$$

Then we consider the case in which the read contains errors inside (u, b) pair. In particular, we consider the case a read covers pair (u, b) and has one error inside this pair. The probability of a read containing more than one error is much smaller so we do not consider these cases. Let p_e be the probability of a read covering (u, b) pair and contains an error. We further assume this error happens uniformly in all the positions in u and position b with probability $\frac{1}{w+1}$, then if the (u, b) pair is at the beginning of the read, the probability of the read contains one error in (u, b) is

$$\begin{aligned} p_{e1} &= \frac{1}{w+1}\{\frac{1-q(1)}{3}q(2)q(3)\dots q(w+1) + q(1)\frac{1-q(2)}{3}q(3)\dots q(w+1) \\ &+ \dots + q(1)q(2)\dots q(w)\frac{1-q(w+1)}{3}\}. \end{aligned}$$

Thus, we can calculate $p_{e2}, p_{e3}, \ldots, p_{e,l-w}$ in the same way and obtain p_e as

$$\begin{aligned} p_e &= \frac{l-w}{L}\frac{1}{l-w}\{p_{e1} + p_{e2} + \ldots + p_{e,l-w}\} \\ &= \frac{1}{L}\frac{1}{w+1}\sum_{i=1}^{l-w}\sum_{j=i}^{j=i+w}\{\prod_{i\le m\le i+w, m\neq j} q(m)\}\frac{1-q(j)}{3}. \end{aligned} \tag{4.3}$$

Let N_e denote the number of such reads, similar as the case with correct (u, b) pair case, we can calculate the probability of $N_e = k$ as

$$P(N_e = k) = \binom{n}{k} p_e^k (1-p_e)^{n-k}.$$

And the expected number of erroneous (u, b) pairs is

$$M_e(k) = \binom{n}{k} p_e^k (1-p_e)^{n-k} L.$$

The analysis in this section is used to determine the threshold T to distinguish the support values of correct (u, b) pairs from erroneous (u, b) pairs. This threshold is used in the error correction algorithm to verify the correctness of the support. We choose T as

$$T = \min\{k | M_c(k) > M_e(k)\}. \tag{4.4}$$

To calculate $M_c(k)$ and $M_e(k)$, we need to know the error profile of the short read data to obtain $q(1), q(2), \ldots, q(l)$. These can be computed from the per-cycle error rates provided by the sequencing platform. For example, the error rates provided by ParticleCall and BayesCall shown in Fig. 2.7.2 can be directly used. In order to process the data sets such as the *E. Coli* data mentioned in Section 3.7.2, we can use the average error rates for each position of the reads as the error profile. The average error rates in the *E. Coli* data set is shown in Fig. 4.1.

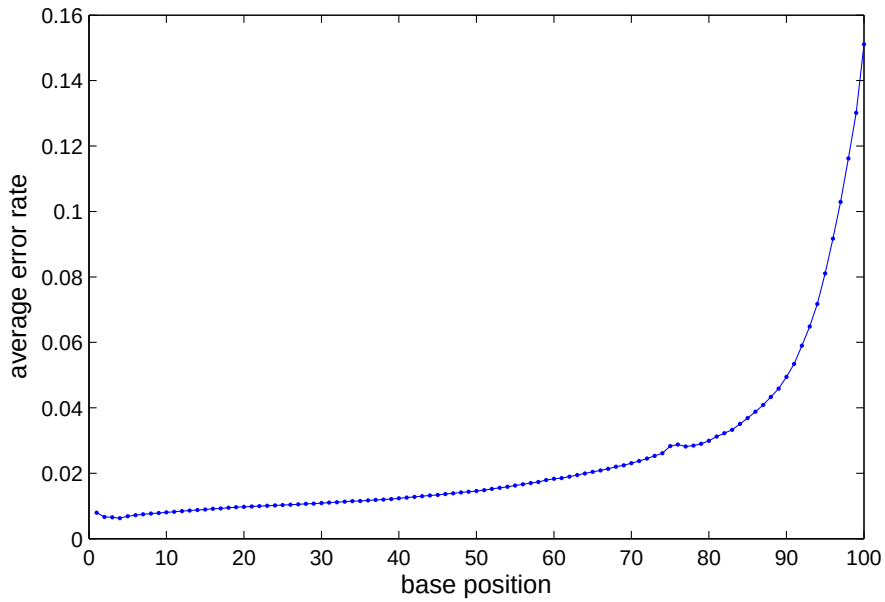

Figure 4.1: *Average error rates in E. Coli data set.*

4.3.2 Choice of optimal support length

In order to calculate the threshold T, the support length w need to be determined beforehand. In current sequencing technology, the read length l is typically around $70 - 120$. The length of w is usually chosen in the range $10 - 20$ in recently developed error correction algorithms. In HiTEC, a more sophisticated approach is used to approximately find an optimal w_m and a set of w values around w_m are tried to achieve the best performance in the error correction algorithm. We combine this approach with a hypothesis testing scheme to obtain optimal w in our error correction algorithm. Denote the average error rate for all the bases in the reads as $\bar{p}$. If w is too large, it is possible that some reads will have errors distributed in a way such that there

are no w consecutive correct positions. In this case the errors in this read is not correctable by the error correction scheme. The expected number of this kind of reads can be calculated by

$$U(w) = \sum_{i=1}^{l} g(i,l)\bar{p}^{i}(1-\bar{p})^{l-i}n,$$

where $g(i,l)$ is the number of possible ways to place i errors in a length l read. It can be calculated using an iterative procedure as shown in [55]. $U(w)$ will decrease when we decrease w. However, if w is too small, the possibility of repetitive regions in the genome with length w increases. In this case, if one read contains pair (u,b) but has an error inside u such that u is changed to v, and v is also in the genome, the b base might be incorrectly changed in the error correction process. The expected number of such reads $D(w)$ can also be calculated using the parameter $\bar{p}$. The theoretical optimal support length w_o can be obtained as

$$w_o = \arg\min_{w}(U(w) + D(w)). \tag{4.5}$$

In practice, we test a set of w values around w_o to obtain optimal w with lowest expected testing error $p_{FP} + p_{FN}$ in hypothesis testing. The details of the hypothesis testing scheme is described in following section.

4.4 Error correction algorithm based on a hypothesis testing scheme

4.4.1 A hypothesis testing scheme to improve the accuracy of support detection

The error correction algorithm basically will look for all the bases with same length w support. The correctness of base b is determined by whether the support value is bigger than threshold T. The suffix array data structure and longest common prefix values can be used to obtain all the bases with same support. We check the set of consecutive positions in the suffix array with longest common prefix value (lcp) larger than w and the $w+1$ position is not the auxiliary symbol \$.

This support detection procedure does not utilize the quality score information provided in the next-generation sequencing data set. In fact, it assumes all the bases in the support sequence are equally reliable, which is not the case in recent sequencing technology. A length w support sequence at the beginning of a read is usually much more reliable than the same length sequence at the end of the read. We can use quality scores to verify the reliability of the read and discard unreliable data to improve the accuracy of the support value calculation. Here we use a hypothesis testing procedure based on quality scores to determine whether a support sequence is correct.

We set the null hypothesis H_0 as the support sequence u is correct and the alternative hypothesis H_1 as u contains errors so it should be disregarded in the support value calculation. We use the summation of the quality values

(the probability of correctness converted from quality score) S as our test statistic. Assume the quality values of the bases inside the length w support sequence u are $q_{u1}, q_{u2}, \ldots, q_{uw}$, then

$$S = q_{u1} + q_{u2} + \ldots + q_{uw}.$$

Intuitively, the higher S is, the more confidence we will have in the correctness of u. So we can choose a parameter θ and set the rejection region of the test as $S < \theta$ and acceptance region as $S > \theta$. For each fixed support length w, we can obtain the conditional pdfs of test statistic S given different hypotheses. Then the test threshold θ can be optimized by minimizing the testing error $p_{FP} + p_{FN}$. We illustrate this procedure using an example in next section.

4.4.2 Error correction algorithm

We can formalize the error correction procedure as Algorithm 4.

The construction of the suffix arrays of R can be efficiently performed by the *libdivsufsort* library [60]. It uses a suboptimal approach but it is proved to be significantly faster and space efficient than the theoretically optimal algorithms in practice.

In step (1.2), we need to obtain the conditional pdfs of the test statistic S given the hypotheses, i.e., $p(S|H_0)$ and $p(S|H_1)$. These two pdfs are in general the same in the same sequencing platform, so we can obtain them from the control lane data.

Algorithm 4 Error correction algorithm

Input: short reads data $r_1, r_2, \ldots, r_n$ and corresponding quality scores for each read base

1. Initialization:

1.1 compute optimal parameters T and w_o using (4.4) and (4.5)

1.2 obtain conditional distribution of the test static S from the control lane data and use them to search optimal θ which minimizes expected testing error.

2. Iterations

repeat

  $count = 0$

  construct R and compute suffix arrays and corresponding lcp

  for each length w support sequence u with number of occurrence $\geq T+1$ **do**

    do hypothesis testing for each u, discard if $S < \theta$

    obtain correct set $B_c = \{b|\text{supp}(u, b) \geq T\}$

    obtain error set $B_e = \{b|\text{supp}(u, b) < T\}$

    for each $b \in B_e$ **do**

      correct b to $a \in B_c$.

    end for

  end for

until $\frac{count}{ln} < 0.0001$

4.5 Experimental results

4.5.1 Parameter calculation

In order to test the performance of the proposed algorithm, we use the *E. Coli* data set from Illumina's HiSeq platform which contains approximately 4×10^6 reads with read length $l = 100$. So the coverage of the data set is around 100. The exact sequence of this *E. Coli* strand is known so we can use it to evaluate the performance of our error correction method.

We use the average quality values of each of the read position to calculate the error profile. The average error rate of all bases is $\bar{p} = 0.025$. Using the error profile and $\bar{p}$, we can obtain $w_o = 17$ and $T = 24$.

For each w value we used in the algorithm, a corresponding θ value need to be determined. We need to obtain two pdfs $p(S|T_0)$ and $p(S|T_1)$ for this sequencing platform. We use the true *E. Coli* sequence and use a short read alignment algorithm to align all the reads to it. Then whether a base in the read is correct is determined according to if it match the base in the exact sequence. We randomly choose 10^7 length w read fragments and plot the histograms of test statistic S, of the correct fragments and erroneous fragments, respectively. These two histograms are used to approximate $p(S|T_0)$ and $p(S|T_1)$. The approximate pdfs for $w = 16$ is shown in Fig 4.2. We search for different w values and find the optimal parameters are $w = 16$ and $\theta = 14.35$ which can minimize $p_{FP} + p_{FN}$.

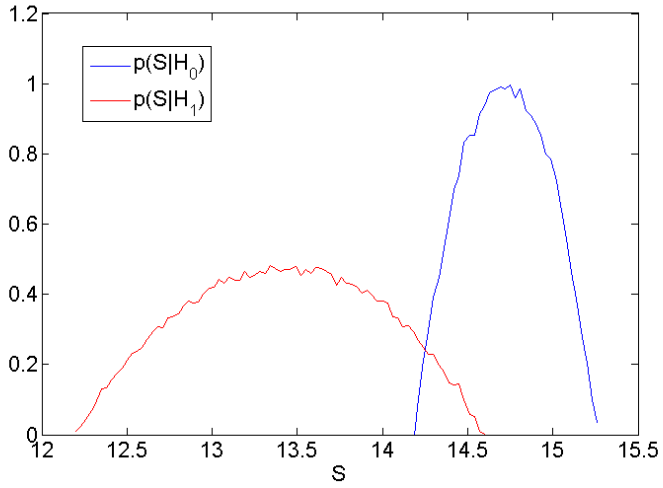

Figure 4.2: *Probability density functions of test statistic S.*

Sequence	Error(%)	this chapter	SHREC	HiTEC
E. coli	3.5	89.6	71.3	83.2
N. Meningitidis	3.9	87.3	69.5	81.1

Table 4.1: Accuracy of different error correction algorithms.

4.5.2 Error correction accuracy

In *de novo* error correction, the accuracy of the algorithm is defined as the ratio between the number of corrected reads and the number of reads contain errors in the original data set. We compare our algorithm with other error correction algorithms HiTEC and SHREC. The comparison result is shown in Table 4.5.2. We can see the accuracy of the error correction can be improved by 5% using our error correction scheme incorporating the quality score information and the hypothesis testing procedure.

4.6 Conclusions and discussion

In this chapter, we consider the error correction problem in *de novo* DNA sequence assembly. We develop an error correction algorithm utilizing the quality information of the reads provided by the next-generation DNA sequencing platform. The algorithm uses suffix array data structure to efficiently identify the repetitive regions between the short reads and correct the erroneous bases by comparing it to the bases in other reads. A hypothesis testing scheme is adopted to improve the support detection accuracy. The test statistic based on the quality scores of certain fragments of the read is used to test the correctness of the fragment. Experimental results show that the proposed error correction algorithm has higher accuracy compared to traditional algorithms.

In this chapter, we use an experimental approach, i.e., plotting the empirical probability density function of $p(S|H_0)$ and $p(S|H_1)$, to calculate the threshold θ used in the hypothesis testing. Whether there exists an analytical form of these two functions can be further discussed to facilitate the usage of the proposed error correction algorithm.

Chapter 5

Inferring Parameters of Gene Regulatory Networks via Particle Filtering

5.1 Background

Gene regulatory networks (GRN) are systems comprising biomolecular components (genes, mRNA, proteins) that interact with each other and through those interactions determine gene expression levels, i.e., determine the rate of gene transcription to mRNA [61–63]. The signals in GRN are carried by molecules. For instance, proteins which enable initiation of the gene transcription to mRNA (so-called *transcription factors*) can be considered as input signals. They bind to the so-called promoter regions adjacent to the regulated gene and, in doing so, enable an RNA Polymerase to perform the transcription. On the other hand, proteins that are translated from the mRNA can be considered as output signals. Some of the created proteins may act as transcription factors themselves and upregulate or downregulate gene expressions, i.e., activate or suppress the transcription process. This creates feedback loops in the network which allow direct or indirect self-regulation. An illustration of a possible segment of a regulatory pathway is shown in Fig. 5.1.

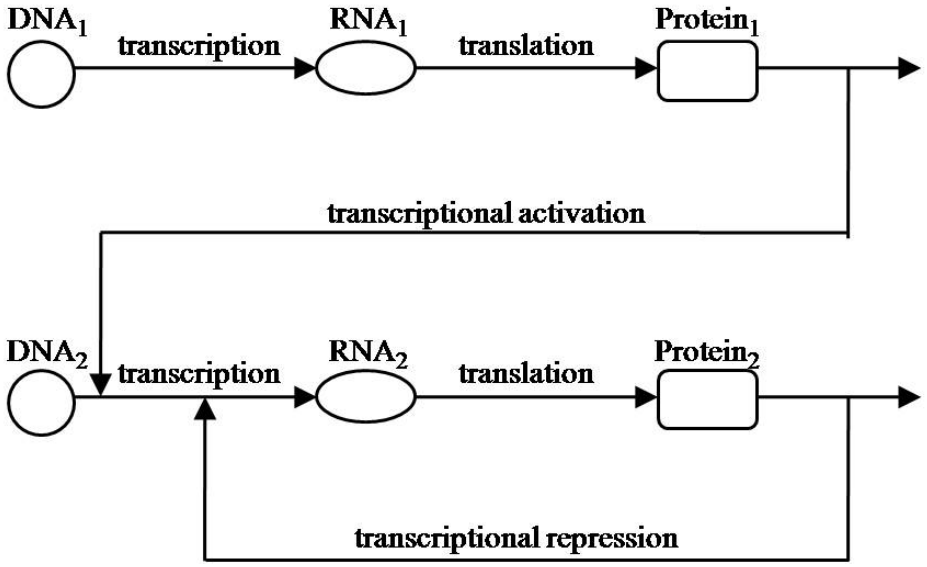

Figure 5.1: *An illustration of a possible segment of a regulatory pathway.*

Recent development of DNA and protein microarrays sparked a surge of interest in studying gene regulatory mechanisms. The excitement is due to the capability of the microarrays to conduct simultaneous tests of an entire genome of an organism. By testing a number of biological samples taken over a period of time, one can track the network dynamics. The experimental advances have been accompanied by the theoretical developments in modeling and computational studies of the networks. Combination of these research efforts provides critical information about the functionality of cells and organisms, reveals mechanisms of genetic diseases, enables optimization of diagnostic techniques and therapies, and provides aid in the process of drug discovery.

To enable the analysis of gene regulatory networks, we need accurate

yet tractable models capturing their dynamical behavior. The molecular interactions in gene regulatory networks are inherently stochastic. For instance, the number of created proteins is a random variable due to thermal fluctuations in a cell which cause promotors to randomly switch between an active and a repressed state. The fluctuations in the number of proteins are enhanced by the protein degradation which is a stochastic process itself. This, along with several other sources of randomness, call for probabilistic modeling of gene regulatory networks. However, a very detailed description of a network may be difficult to analyze and often requires considerable computational efforts. Hence, several models with varying degree of accuracy and complexity have been proposed. These models rely on representations via chemical master and chemical Langevin equations [64, 66, 67], ordinary differential equations [68, 69], Bayesian [70, 71] and Boolean [63, 72] networks. Having selected one of the above models, we are interested in finding its structure and parameters that provide the best explanation of the experimental data. This requires further computational studies and opens up questions related to, e.g., stability and control of the network. However, inference problems in gene regulatory networks are often challenging, and the difficulty of a problem increases with the complexity of the model and the size of the network.

In this chapter, we consider models of GRN based on chemical master equations, and study the problem of estimating stochastic rate constants therein. Such models provide the most precise description of the network processes; however, they are also computationally the most demanding. We

limit our focus on small-sized networks with a known structure but unknown rate constants. We approximate a chemical master equation by a related chemical Langevin equation [17], and employ a particle filter with the Markov Chain Monte Carlo move step to solve the rate estimation problem. Simulation studies demonstrate that the proposed technique outperforms previously considered methods while being computationally more efficient. Dynamic behavior of gene regulatory networks averaged over a large number of cells can be modeled by ordinary differential equations. For this scenario, we compute an approximation to the Cramer-Rao lower bound on the mean-square error of estimating reaction rates and demonstrate that, when the number of unknown parameters is small, the proposed particle filter can be nearly optimal.

The chapter is organized as follows. Section 5.3 describes the chemical master equation model of a gene regulatory network and its approximation by a chemical Langevin equation. Section 5.4 presents the particle filtering algorithm for the estimation of the stochastic rate constants, and compares its performance with prior work. In Section 5.5, a deterministic model based on ordinary differential equations is described, and the Cramer-Rao lower bound on the performance of estimating rate constants is computed. Finally, we conclude the chapter in Section 5.6.

5.2 Contributions

In this chapter, a particle filter with Markov Chain Monte Carlo move step is employed for the estimation of reaction rate constants in gene regula-

tory networks modeled by chemical Langevin equations. Simulation studies demonstrate that the proposed technique outperforms previously considered methods while being computationally more efficient. Dynamic behavior of gene regulatory networks averaged over a large number of cells can be modeled by ordinary differential equations. For this scenario, we compute an approximation to the Cramer-Rao lower bound on the mean-square error of estimating reaction rates and demonstrate that, when the number of unknown parameters is small, the proposed particle filter can be nearly optimal.

5.3 Models Based on Chemical Master and Chemical Langevin Equations

Consider a GRN comprising N molecular components. The network variables are the numbers of the molecules of each of the N species; generally, we are interested in the temporal changes of these variables. Denote the number of molecules of the i^{th} network component at time t by $x_i(t)$; for convenience, collect the $x_i(t)$ into a vector $X(t)$, i.e., denote $X(t) = [x_1(t) \ \ldots \ x_N(t)]^T$. Molecular reactions in a GRN are subject to significant spontaneous fluctuations. Consequently, the numbers of the molecular species $x_i(t)$ are inherently stochastic processes. We can model $X(t)$ as a Markov process with discrete states, where the time evolution of the state probabilities $P(X,t)$ is given by the chemical master equation, $\frac{\partial P(X,t)}{\partial t} =$

$$\sum_{m=1}^{M} \left[a_m(X - \mathcal{V}_m) P(X - \mathcal{V}_m, t) - a_m(X) P(X,t) \right]. \tag{5.1}$$

In (5.1), M denotes the total number of reactions that are possible within the network (i.e., the number of the so-called *reaction channels*), $\mathcal{V}_m = [v_{m1} \;\; v_{m2} \;\; \ldots \;\; v_{mN}]^T$ is the vector describing change in the number of molecules of each of the N species due to the reaction in the m^{th} reaction channel (e.g., v_{mi} is the change, either positive or negative, in the number of molecules of the i^{th} network component due to the reaction in the m^{th} channel). Moreover, $a_m(\cdot)$ in (5.1) is the so-called propensity function, i.e., $a_m(\cdot)dt$ is the probability that during time interval $(t, t+dt)$ there is a reaction in the m^{th} channel. The propensity function can further be expressed as $a_m(X(t)) = c_m h_m(X(t))$, where $c_m dt$ is the probability that one reaction takes place in $(t, t+dt)$ and $h_m(X(t))$ denotes the number of possible simultaneous reactions[1]. The chemical master equation is often used to simulate the Markov process $X(t)$ and enable computational studies of GRN. To this end, one may employ various stochastic simulation algorithms, originally proposed by Gillespie [64].

The model (5.1) provides a very accurate description of the network dynamics [64]. However, since it tracks individual discrete events, it is often cumbersome for practical purposes. For instance, relying on (5.1) to infer the parameters of the network (i.e., the stochastic rate constants c_m) may in principle be possible [73]; however, it is computationally rather intensive to do so. Therefore, simplified network models are desirable. Under certain assumptions (e.g., large $x_i(t)$, small dt), we may approximate (5.1) by the

[1]The coefficients c_m are often referred to as the stochastic rate constants. The function $h_m(X(t))$ counts all possible combinations of individual molecules that may lead to a reaction in the m^{th} channel.

chemical Langevin equation, $X(t+dt) - X(t) =$

$$\sum_{m=1}^{M} \left[\mathcal{V}_m a_m(X(t))dt + \mathcal{V}_m \sqrt{a_m(X(t))dt} \mathcal{N}_m(0,1) \right], \tag{5.2}$$

where $\mathcal{N}_m(0,1)$ denote zero-mean, unit-variance, independent, identically distributed (iid) Gaussian random variables. By collecting vectors $\mathcal{V}_m$ into a stoichiometry matrix $S = [\mathcal{V}_1 \;\; \mathcal{V}_2 \;\; \ldots \;\; \mathcal{V}_M]$, we can write (5.2) as

$$X(t+dt) - X(t) = S\mathbf{a}(X(t))dt + (SA(X(t))S^T)^{1/2} dW, \tag{5.3}$$

where dW denotes an M-dimensional Wiener process, vector $\mathbf{a}(X(t))$ is defined as

$$\mathbf{a}(X(t)) = [a_1(X(t)) \;\; a_2(X(t)) \;\; \ldots \;\; a_M(X(t))]^T,$$

and where

$$A(X(t)) = \text{diag}\,\{a_1(X(t)), a_2(X(t)), \ldots, a_M(X(t))\}.$$

We should point out that while the chemical Langevin equation (5.2) may be used as a network model for the purpose of parameter estimation, in general it is not sufficiently accurate to provide reliable simulations of the network dynamics. To conduct computational studies of a GRN, we still need to model them using stochastic simulation algorithms.

Let us write the chemical Langevin equation (5.3) using the notation typically encountered in the literature on stochastic differential equations,

$$X(t+dt) - X(t) = \mu(X(t), \boldsymbol{\theta})dt + \sigma(X(t), \boldsymbol{\theta})dW, \tag{5.4}$$

where $\mu(X(t), \boldsymbol{\theta}) = S\mathbf{a}(X(t))$ denotes the drift, and $\sigma(X(t), \boldsymbol{\theta}) = (SA(X(t))S^T)^{1/2}$ is the diffusion, and $\boldsymbol{\theta}$ is the vector of (generally unknown) parameters (i.e., the elements of $\boldsymbol{\theta}$ are the stochastic rate constants c_i). Our goal is to infer $\boldsymbol{\theta}$ from $X(t)$ observed at discrete time instances $t_i = i\Delta, 1 \leq i \leq L$, where L denotes the total number of observations. Assuming zero-mean Gaussian measurement noise with covariance matrix Σ, the collected observations have normal distribution of the form

$$y_i = y(i\Delta) \sim \mathcal{N}(X(i\Delta), \Sigma).$$

In [74], the authors find the best linear-model fit to the data presumed to be generated by (5.4), and then infer parameters based on the derived linear model. In [75, 76], the use of statistical mechanics tools for the estimation of the parameters of a network modeled by (5.4) was considered. In [77, 78], a Markov Chain Monte Carlo (MCMC) algorithm was employed to infer the network parameters. This approach provides sound estimate of the parameters but it requires a very high computational effort. As an alternative, we propose the use of a particle filter with an MCMC move step. This we describe in the next section.

5.4 Particle Filter with Markov Chain Monte Carlo move step

We consider Bayesian approaches to inferring the unknown parameters in $\boldsymbol{\theta}$, which is treated as a random vector with a prior $p(\boldsymbol{\theta})$. Specifically, we rely on particle filtering methods to infer the posterior distribution

$p(\boldsymbol{\theta}|y_{1:N})$, and then find the estimate $\hat{\boldsymbol{\theta}}$ as the conditional mean of $p(\boldsymbol{\theta}|y_{1:N})$. Here $y_{1:N} = \{y_1, y_2, \ldots, y_N\}$ denotes the set of observations collected in the interval $[\Delta, N\Delta]$, where Δ denotes the sampling period and N denotes the total number of observations (e.g., y_n is the noisy observation collected at time $n\Delta$). The desired posterior distribution can be expressed as

$$p(\boldsymbol{\theta}|y_{1:N}) = \int p(x_{1:N}, \boldsymbol{\theta}|y_{1:N}) dx_{1:N},$$

where $x_{1:N} = \{x_1, x_2, \ldots, x_N\}$ denotes the set of points of the process $X(t)$ corresponding to the observations in $y_{1:N}$ (e.g., $x_n = X(n\Delta)$), and $p(x_{1:N}, \boldsymbol{\theta}|y_{1:N})$ is given by

$$p(x_{1:N}, \boldsymbol{\theta}|y_{1:N}) \propto p(y_{1:N}|x_{1:N}, \boldsymbol{\theta}) p(x_{1:N}|\boldsymbol{\theta}) p(\boldsymbol{\theta}). \tag{5.5}$$

To evaluate (5.5), one needs to compute the joint density

$$p(x_{1:N}|\boldsymbol{\theta}) = p(x_N|x_{N-1}, \boldsymbol{\theta}) \ldots p(x_2|x_1, \boldsymbol{\theta}) p(x_1|\boldsymbol{\theta}).$$

In general, however, the transition densities

$$p(x_{n+1}|x_n, \boldsymbol{\theta}) = p(X((n+1)\Delta)|X(n\Delta), \boldsymbol{\theta})$$

for the process (5.4) are not available in a closed form. The stochastic differential equation (5.4) can be discretized using the Euler-Maruyama scheme as

$$x_{n+1} = x_n + \mu(x_n, \boldsymbol{\theta})\Delta + \sigma(x_n, \boldsymbol{\theta})\delta W,$$

where δW denotes a zero-mean Gaussian distribution with covariance ΔI, and I denotes the identity matrix. Hence the transition density $p(x_n|x_{n-1}, \boldsymbol{\theta})$ can

be approximated by a Gaussian distribution with mean $x_n + \mu(x_n, \boldsymbol{\theta})\Delta$ and covariance $\sigma(x_n, \boldsymbol{\theta})(\sigma(x_n, \boldsymbol{\theta}))^T\Delta$. However, the Euler-Maruyama approximation of the transition density is accurate only when Δ is small. If the sampling period is not sufficiently small, one can introduce the so-called missing values $z_{1:m} = \{z_1, z_2, \ldots, z_m\}$ which emulate the diffusion process between x_n and x_{n+1} (a distinct set of missing values is introduced for each n). The number of augmented missing values m is chosen such that the Euler-Maruyama approximation of the transition density between z_k and z_{k+1} is accurate, i.e., m is chosen such that $p(z_{k+1}|z_k, \boldsymbol{\theta})$ can be closely approximated by a Gaussian distribution. It is straightforward to show that

$$\widetilde{\pi}(z_j|z_{j-1}, \boldsymbol{\theta}_{n-1}, y_n) = \mathcal{N}(z_{j-1} + \psi\frac{\Delta}{m}, \gamma\frac{\Delta}{m}), \tag{5.6}$$

where $\psi = \mu + \beta(\beta\Delta_j + \Sigma)^{-1}(y_n - [x_{n-1} + \mu\Delta_j])$, $\gamma = \beta - \beta(\beta\Delta_j + \Sigma)^{-1}\beta^T\frac{\Delta}{m}$, $\mu = S\mathbf{a}(x_{n-1})$, $\beta = SA(x_{n-1})S^T$, $\Delta_j = (m - j + 1)\frac{\Delta}{m}$, and Σ denotes the covariance matrix of the measurement noise.

Introduction of the missing values enables propagating (5.5) by means of a particle filter, where the filter relies on a Gaussian importance density (5.6). A simple sequential importance resampling (SIR) scheme provides asymptotically consistent estimates, i.e., the approximation converges to the true value of the parameters as the number of particles grows. However, the SIR scheme often suffers from sample impoverishment and, therefore, has weak performance. To improve the sample diversity and the performance of the particle filter, we employ the importance sampling scheme with an MCMC move

step. Specifically, we use the Metropolis-Hastings algorithm to decide whether a resampled particle will be accepted or not. For implementation details, we refer the reader to the formal algorithm given below:

1. (*Initialization*) Set $n = 1$. Draw $\{\boldsymbol{\theta}_n^i, x_n^i\}_{i=1}^{N_s}$ from the prior density $\pi(\boldsymbol{\theta})\pi(x_n)$. Assign particle weights $\omega_n^i = \pi(y_n|x_n^i, \boldsymbol{\theta}_n^i)$, for $i = 1, 2, ..., N_s$, and normalize them.

2. (*Iterations*) For $n \geq 2$:

2.1 For $i = 1, ..., N_s$, draw missing data $\{z_k^i\}_{k=1}^m$ from an importance density

$$q(z_1, \ldots, z_m|\boldsymbol{\theta}_{n-1}^i, x_{n-1}^i, y_n)$$

obtained using the Euler approximation as

$$\pi(z_1|x_{n-1}^i)\prod_{j=2}^{m}\widetilde{\pi}(z_j^i|z_{j-1}^i, \boldsymbol{\theta}_{n-1}^i, y_n),$$

where $\widetilde{\pi}(z_j^i|z_{j-1}^i, \boldsymbol{\theta}_{n-1}^i, y_n) =$

$$\mathcal{N}(z_{j-1}^i + \psi\frac{\Delta}{m}, \gamma\frac{\Delta}{m}),$$

$\psi = \mu + \beta(\beta\Delta_j + \Sigma)^{-1}(y_n - [x_{n-1}^i + \mu\Delta_j])$, $\gamma = \beta - \beta(\beta\Delta_j + \Sigma)^{-1}\beta^T\frac{\Delta}{m}$, $\mu = S\mathbf{a}(x_{n-1}^i)$, $\beta = SA(x_{n-1}^i)S^T$, $\Delta_j = (m - j + 1)\frac{\Delta}{m}$.

Set $x_n^i = z_m^i$ and update the particle weights as

$$\begin{aligned}\omega_n^i &= \omega_{n-1}^i\pi(y_n|x_n^i, \boldsymbol{\theta}_{n-1}^i)\pi(z_1^i|x_{n-1}^i, \boldsymbol{\theta}_{n-1}^i) \\ &\times \frac{\prod_{j=2}^{m}\pi(z_j^i|z_{j-1}^i, \boldsymbol{\theta}_{n-1}^i)}{q(z_1, ..., z_m|\boldsymbol{\theta}_{n-1}^i, x_{n-1}^i, y_n)}.\end{aligned}$$

2.2 (*Normalization*) Normalize the weights ω_n^i, and compute $N_{\text{eff}} = \frac{1}{\sum_{k=1}^{N_s} (\omega_n^k)^2}$.

2.3 (*Resampling*) If $N_{\text{eff}} < N_{\text{threshold}}$,

$\{\boldsymbol{\theta}_n^{i*}, x_n^{i*}, \frac{1}{N_s}\}_{i*=1}^{N_s} = \text{Resample}(\{\boldsymbol{\theta}_n^i, x_n^i, \omega_n^i\}_{i=1}^{N_s})$.

2.4 (*Resample move*) If resampling is performed in step 2.3, then for $i = 1, ..., N_s$:

(a) Draw a candidate $\boldsymbol{\theta}_*$ from a kernel density $\mathcal{K}(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}_n^i, h_{opt}S)$, where S is the empirical covariance of $\boldsymbol{\theta}$ in the previous step and h_{opt} is the smoothing parameter.

(b) Draw missing data $\{z_{k*}^i\}_{k*=1}^m$ from an importance density

$$q(z_{1*}, ..., z_{m*}|\boldsymbol{\theta}_*, x_{n-1}^i, y_n)$$

and set $x_{n*}^i = z_{m*}^i$.

(c) Calculate the Metropolis-Hastings acceptance rate

$$\begin{aligned} \alpha &= \frac{\pi(y_n|x_{n*}^i, \boldsymbol{\theta}_*)\pi(z_{1*}|x_{n-1}^i, \boldsymbol{\theta}_*)}{\pi(y_n|x_n^i, \boldsymbol{\theta}_n^i)\pi(z_1^i|x_{n-1}^i, \boldsymbol{\theta}_{n-1}^i)} \\ &\times \frac{\prod_{j=2}^m \pi(z_{j*}^i|z_{(j-1)*}^i, \boldsymbol{\theta}_*)}{\prod_{j=2}^m \pi(z_j^i|z_{j-1}^i, \boldsymbol{\theta}_{n-1}^i)} \frac{q(z_1, ..., z_m|\boldsymbol{\theta}_{n-1}^i, x_{n-1}^i, y_n)}{q(z_{1*}, ..., z_{m*}|\boldsymbol{\theta}_*, x_{n-1}^i, y_n)}. \end{aligned}$$

(d) Set $(\boldsymbol{\theta}_n^i, x_n^i) = (\boldsymbol{\theta}_*, x_{n*}^i)$ with prob. $\min\{1, \alpha\}$.

5.4.1 Computational study of a viral infection network

We demonstrate the performance of the proposed algorithm on a viral infection network previously studied in [81, 83]. The network comprises 6 reaction channels,

$$
\begin{aligned}
R_1: &\quad DNA + P \rightarrow V,\\
R_2: &\quad DNA \rightarrow RNA + DNA,\\
R_3: &\quad RNA \rightarrow \emptyset,\\
R_4: &\quad RNA \rightarrow DNA + RNA,\\
R_5: &\quad RNA \rightarrow P + RNA,\\
R_6: &\quad P \rightarrow \emptyset,
\end{aligned}
$$

where P denotes viral protein molecules, and V denotes synthesized viral cells. Reaction R_1 is the process of producing viral cells from the viral DNA and protein. Reactions R_2 and R_5 are the transcription and translation process of the viral genes, respectively. Reaction R_4 models replication of a viral RNA template into a viral DNA.

For the purpose of parameter estimation, we assume that the above network evolves according to (5.3). However, the network is simulated via the Gillespie's algorithm, with the rate constants set to $[c_1\ c_2\ c_3\ c_4\ c_5\ c_6]^T = [11.25 \times 10^{-3}\ 0.25\ 0.5\ 1\ 2\ 1]^T$. We refer to the proposed particle filtering algorithm with MCMC move step as Alg.1, and employ it to estimate parameters in this network. The performance of the Alg.1 is compared to the MCMC method proposed in [78], denoted for convenience as Alg.2. Alg.1 is performed with $N_s = 2 \times 10^4$, $m = 15$, and the resampling threshold $N_{\text{threshold}} = N_s/2$. Both algorithms use $N = 40$ noisy observations of the network states, and employ the same initial sample distribution. The log-values of the parameters

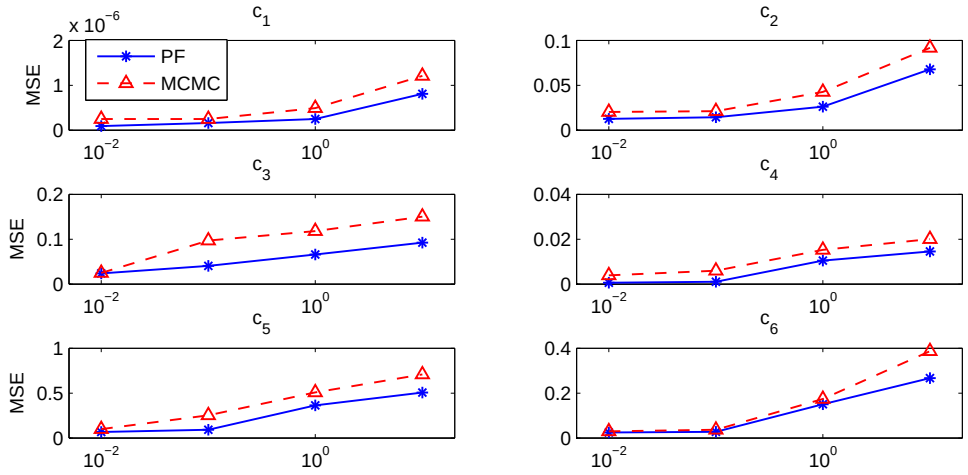

Figure 5.2: *The mean-square-error performance comparison between Alg.1 (particle filter) and Alg.2 (MCMC) as a function of the variance of the observation noise* σ^2 *(*$N_s = 2 \times 10^4$*,* $m = 15$*,* $N = 40$*,* $\Sigma = \sigma^2 I$*).*

$\log(\boldsymbol{\theta}_i)$ are initialized from the uniform distribution $\mathcal{U}(-4, 2)$, and the noise variance is assumed to be known. [Note that even though c_1 does not belong to the initialization range, the proposed technique accurately infers its value.] Fig. 5.4.1 compares the mean-square-error of estimating the parameters of the viral infection network using Alg.1 and Alg.2, obtained by performing 150 simulation runs. Clearly, the proposed Alg.1 outperforms Alg.2, while being roughly 5 times faster – the average running time of Alg.1 is 1030 seconds, while the average running time of Alg.2 is 5500 seconds (simulations in Matlab).

5.4.2 Computational Study of Procaryotic Regulation

In this subsection, we illustrate the performance of the proposed algorithm when employed for estimating reaction rates in a network with 12 parameters. In particular, we consider estimation of the reaction rates in a GRN model of procaryotic auto regulation. The system is characterized by the following 12 reactions [78]:

$$
\begin{aligned}
R_1: &\quad P_1 + DNA_1 \to P \cdot DNA_1,\\
R_2: &\quad P \cdot DNA_1 \to P_1 + DNA_1,\\
R_3: &\quad P_1 + DNA_2 \to P \cdot DNA_2,\\
R_4: &\quad P \cdot DNA_2 \to P_1 + DNA_2,\\
R_5: &\quad DNA_1 \to DNA_1 + RNA_1,\\
R_6: &\quad RNA_1 \to RNA_1 + P_1,\\
R_7: &\quad DNA_2 \to DNA_2 + RNA_2,\\
R_8: &\quad RNA_2 \to RNA_2 + P_2,\\
R_9: &\quad RNA_1 \to \emptyset,\\
R_{10}: &\quad RNA_2 \to \emptyset,\\
R_{11}: &\quad P_1 \to \emptyset,\\
R_{12}: &\quad P_2 \to \emptyset.
\end{aligned}
$$

Reactions $R_1 \sim R_4$ represent the reversible processes of repressor protein P_1 binding to DNA_1 and DNA_2. Reactions $R_5 \sim R_8$ are the transcription and translation processes of genes DNA_1 and DNA_2. Reactions $R_9 \sim R_{12}$ represent the degradation process of proteins and mRNAs in the system. The state vector X collects the numbers of components DNA_1, DNA_2, RNA_1, RNA_2, P_1, and P_2, and hence is a 6-dimensional state vector.

Similar to the study of the viral infection network in the previous subsection, to infer the reaction rates we assume that the above network evolves according to (5.3). However, the network is simulated via the Gillespie's algo-

		Alg.1	Alg.2(1)	Alg.2(2)
c_1	0.08	0.0707	0.0443	0.0869
c_2	0.82	0.8219	0.6726	0.7134
c_3	0.09	0.0597	0.1121	0.0650
c_4	0.9	0.5625	1.3913	0.5943
c_5	0.25	0.3283	0.1826	0.2862
c_6	0.1	0.1195	0.5800	0.0469
c_7	0.35	0.2875	0.9009	0.2561
c_8	0.3	0.4167	0.8943	0.3577
c_9	0.1	0.1197	0.1573	0.0985
c_{10}	0.1	0.1432	0.5097	0.2943
c_{11}	0.12	0.1178	1.2766	0.0984
c_{12}	0.1	0.1384	0.1669	0.1232
time(s)		5.9×10^4	5.3×10^4	2.5×10^5

Table 5.1: True and estimated parameters for the two algorithms. Alg.2(1) employs 2×10^5 MCMC iterations and Alg.2(2) employs 10^6 iterations.

rithm. In particular, we generate $N = 30$ noisy observations y_n, $1 \leq n \leq 30$, where the measurement noise is Gaussian with $\sigma^2 = 1$ (i.e., the noise variance matrix is $\Sigma = I$). The particle filter (our Alg.1) is performed with $N_s = 2\times10^5$, $m = 20$, and the resampling threshold $N_{\text{threshold}} = N_s/5$. The log-values of the parameters $\log(\boldsymbol{\theta}_i)$ are initialized from the uniform distribution $\mathcal{U}(-5, 1)$.

The reaction rates are inferred as the mean values of the distributions estimated by the particle filter. True values of the parameters and their estimates are shown in Table 5.4.2. When Alg. 2 is performed with $m = 20$ and 2×10^5 MCMC iterations with a 3×10^4 burn-in period, the runtime of Alg.1 and Alg.2 is comparable but the former is significantly more precise than the latter. In order to achieve similar performance, Alg.2 requires significantly

higher complexity (10^6 MCMC iterations with a 3×10^4 burn-in period).

5.5 A Deterministic Model of Gene Regulatory Networks

In reaction systems where both the number of molecules and the system volume are large, due to averaging the system dynamics can be described by a deterministic model. The same applies to modeling the dynamic behavior of a gene regulatory network averaged over a large number of cells. A deterministic model based on ordinary differential equations (ODE) is of the form [17, 65]

$$\frac{d\mathbf{x}(t)}{dt} = S\mathbf{a}(\mathbf{x}(t), \boldsymbol{\theta}), \tag{5.7}$$

where $\mathbf{x}(t)$ comprises real-valued and deterministic variables. On the other hand, the observation process is assumed to be corrupted by a Gaussian noise and hence the measurements are given by

$$y(t) = \mathbf{x}(t) + \mathbf{v}(t). \tag{5.8}$$

Typically, observations are collected at discrete time instances $t_i = i\Delta, 1 \leq i \leq L$, where L denotes the total number of observations. Therefore,

$$y(i\Delta) = \mathbf{x}(i\Delta) + \mathbf{v}(i\Delta),\ 1 \leq i \leq L, \tag{5.9}$$

where $E\{\mathbf{v}(i\Delta)v(j\Delta)^T\} = \sigma_{\mathbf{v}}^2 I_N \delta_{ij}$.

To facilitate a simple estimation procedure, (5.7) can be discretized as

$$\mathbf{x}((i+1)\Delta) - \mathbf{x}(i\Delta) = \Delta \cdot S\mathbf{a}(\mathbf{x}(i\Delta), \boldsymbol{\theta}), i = 1, \ldots, L-1. \tag{5.10}$$

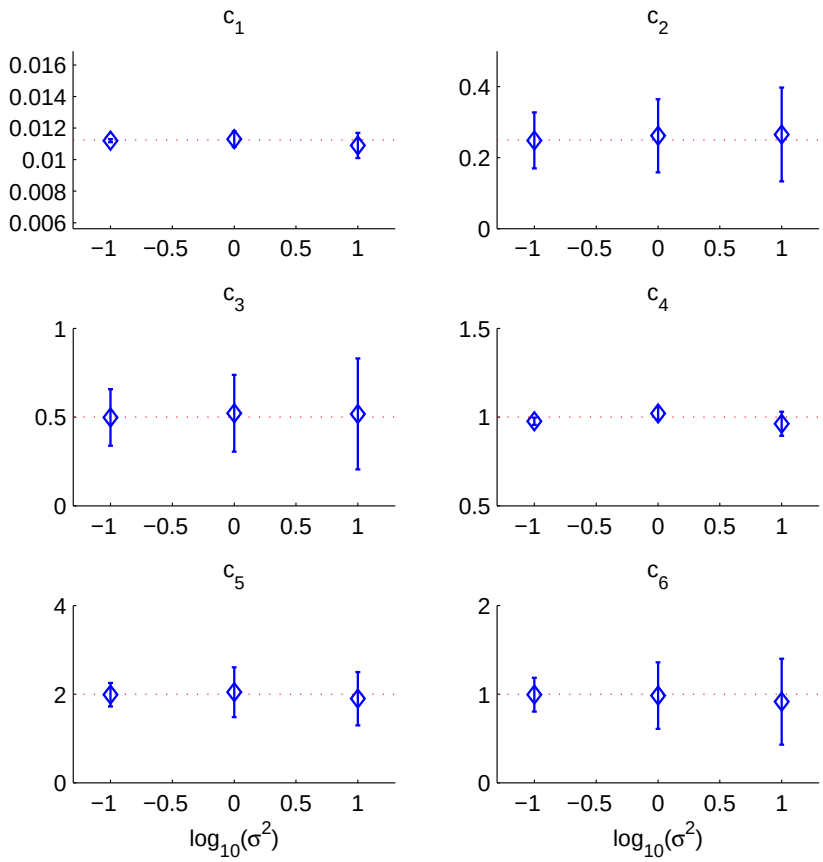

Figure 5.3: *The mean and standard error of the particle filter estimator for the inference of reaction rates in a viral infection network, shown as a function of the variance of the observation noise (the number of particles used is* $N_s = 10^4$*, performance is averaged over* 150 *simulation runs).*

It was pointed out in [80] that, under appropriate conditions, discretization induces smaller error than the measurement noise. In general, we assume that $\Delta \ll \sigma^2$. To estimate the unknown parameters $\boldsymbol{\theta}$ in (5.10), we employ the particle filtering with MCMC step (i.e., Algorithm 1 in Section 5.4). Since the state transitions in the model (5.10) are deterministic (and not random, as in (5.3)), some of the steps of Alg.1 simplify. In particular, Step 2.1 of Algorithm 1 can be simplified in the following way: for each particle, instead of drawing a series of missing data $\{z_k^i\}_{k=1}^m$ from an importance distribution, we deterministically generate them from the previous state x_{n-1}^i as

$$z_1^i = x_{n-1}^i + \frac{\Delta}{m} \cdot S\mathbf{a}(\mathbf{x}_{n-1}^i, \boldsymbol{\theta}_{n-1}^i),$$

$$z_k^i = z_{k-1}^i + \frac{\Delta}{m} \cdot S\mathbf{a}(\mathbf{z}_{k-1}^i, \boldsymbol{\theta}_{n-1}^i), \quad k = 2, \ldots, m$$

and $x_n^i = z_m^i$. The weights updating equation becomes

$$\omega_n^i = \omega_{n-1}^i \pi(y_n | x_n^i, \boldsymbol{\theta}_{n-1}^i).$$

Moreover, the Metropolis-Hastings acceptance rate is simplified to

$$\alpha = \frac{\pi(y_n | x_{n*}^i, \boldsymbol{\theta}_*)}{\pi(y_n | x_n^i, \boldsymbol{\theta}_n^i)}.$$

Other steps of Algorithm 1 remain unchanged.

5.5.1 Cramer-Rao lower bound on the mean-square error of estimating reaction rates

Mean-square error of any estimation procedure can be bounded below by the Cramer-Rao lower bound (CRLB) [79]. In this section, we compute the

CRLB on the estimation of reaction rates in the network described by (5.7), (5.9). Collect the observations $y(i\Delta)$, $i = 1, 2, \ldots, L$, into a vector

$$\mathbf{y} = [y(\Delta)^T \; y(2\Delta)^T \; \ldots \; y(L\Delta)^T]^T.$$

The Cramer-Rao lower bound on the minimum mean-square error of estimating a parameter $\boldsymbol{\theta}_i$ given $\mathbf{y}$ is computed as

$$E\left(\hat{\boldsymbol{\theta}}_i - \boldsymbol{\theta}_i\right)^2 \geq [F^{-1}]_{ii}, \tag{5.11}$$

where the Fisher information matrix F is given by the negative of the expected value of the Hessian matrix of $\log p_{\mathbf{y}|\boldsymbol{\theta}}(\mathbf{y})$,

$$F_{ij} = -E_{\mathbf{y}} \frac{\partial^2}{\partial\theta_i \partial\theta_j} \log p_{\mathbf{y}|\boldsymbol{\theta}}(\mathbf{y}). \tag{5.12}$$

From (5.9), it follows that $\mathbf{y}$ is a Gaussian vector with mean $\bar{y}(i\Delta) = \mathbf{x}(i\Delta)$ and covariance $R = \sigma_{\mathbf{v}}^2 I_{NL}$. Thus we have

$$p_{\mathbf{y}|\boldsymbol{\theta}}(\mathbf{y}) = \frac{1}{\sqrt{(2\pi)^{NL}|R|}} \exp[-\frac{1}{2}(\mathbf{y} - \bar{\mathbf{y}})^T R^{-1}(\mathbf{y} - \bar{\mathbf{y}})].$$

Following a similar derivations in [82], we obtain

$$F_{ij} = \left(\frac{\partial\bar{\mathbf{y}}}{\partial\theta_i}\right)^T R^{-1} \left(\frac{\partial\bar{\mathbf{y}}}{\partial\theta_j}\right) + \frac{1}{2}\text{tr}\left\{R^{-1}\frac{\partial R}{\partial\theta_i}R^{-1}\frac{\partial R}{\partial\theta_j}\right\}. \tag{5.13}$$

Since R is known, $\partial R/\partial\theta_i = 0$, and thus

$$F_{ij} = \left(\frac{\partial\bar{\mathbf{y}}}{\partial\theta_i}\right)^T R^{-1} \left(\frac{\partial\bar{\mathbf{y}}}{\partial\theta_j}\right). \tag{5.14}$$

Therefore, only $\partial\bar{\mathbf{y}}/\partial\theta_i$ is needed to evaluate F_{ij}. From (5.8), it follows that $\bar{y}(t) = x(t)$. Moreover, since $a_m(\mathbf{x}(t)) = \theta_m h_m(\mathbf{x}(t))$, we can write

$$\mathbf{a}(\bar{y}(t)) = \text{diag}\,\{\boldsymbol{\theta}\}H(\bar{y}(t)),$$

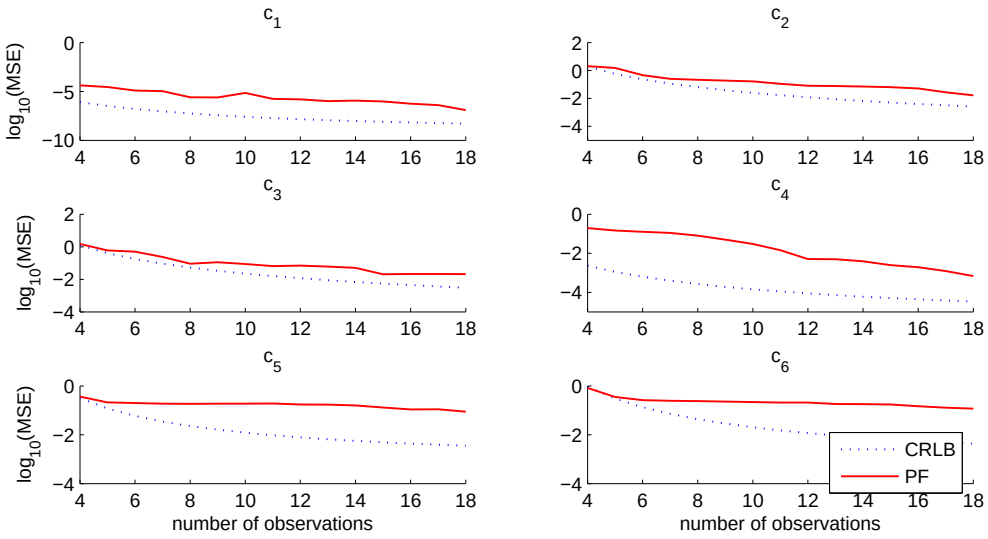

Figure 5.4: *The CRLB and the average mean-square error of the particle filtering algorithm (the number of particles $N_s = 10^4$, noise covariance matrix $\Sigma = I$).*

where $H(\bar{y}(t)) = [h_1(\bar{y}(t))\ h_2(\bar{y}(t)) \dots h_M(\bar{y}(t))]^T$. Taking derivatives of both side of (5.10), we obtain

$$\begin{aligned} \frac{\partial \bar{y}((i+1)\Delta)}{\partial \theta_m} &= \frac{\partial \bar{y}(i\Delta)}{\partial \theta_m} + \Delta \cdot S \cdot E_i H(\bar{y}(i\Delta)) \\ &+ \Delta \cdot S \mathrm{diag}\,\{\theta\} \frac{\partial H(\bar{y}(i\Delta))}{\partial \theta_m}, \end{aligned} \tag{5.15}$$

where E_i denotes the $M \times M$ matrix with all zero entries except the (i,i) entry which is equal to 1.

Notice that $\partial h_i(\bar{y}(i\Delta))/\partial \theta_m$ are functions of $\bar{y}(i\Delta)$ and $\partial h_i(\bar{y}(i\Delta))/\partial \theta_m$; therefore, we can recursively calculate $\partial \bar{y}((i+1)\Delta)/\partial \theta_m$ from $\partial \bar{y}(i\Delta)/\partial \theta_m$ and $\bar{y}(i\Delta)$. The value of $\bar{y}(t)$s can be obtained by numerically solving (5.7) (e.g.,

using *Mathematica*). This enables computation of $\partial\bar{\mathbf{y}}/\partial\theta_m$ and, therefore, the desired CRLB. [Note: the CRLB computed in this section assumes the discretized model (5.10); as $\Delta \to 0$, it approaches the true bound on estimating $\boldsymbol{\theta}$ in (5.7).]

5.5.2 Computational study of a viral infection network

We illustrate the performance of the particle filter and compare it with the computed CRLB for the case of the viral infection network studied in Section 3.1. We assume that the network evolves according to the ODE model described in this section. The rate constants associated with reactions are, as before, $[c_1\ c_2\ c_3\ c_4\ c_5\ c_6]^T = [11.25 \times 10^{-3}\ 0.25\ 0.5\ 1\ 2\ 1]^T$. We apply the modified version of Algorithm 1 described in this section to estimate the rate constants, and evaluate the corresponding CRLB. Note that, in this example, the stoichiometry matrix is given by

$$S = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & -1 \end{bmatrix},$$

and hence

$$\frac{\partial H(\bar{y}(i\Delta))}{\partial \mathbf{c}_m} = \begin{bmatrix} \frac{\partial \bar{y}_3(i\Delta)}{\partial \mathbf{c}_m}\bar{y}_4(i\Delta) + \bar{y}_3(i\Delta)\frac{\partial \bar{y}_4(i\Delta)}{\partial \mathbf{c}_m} \\ \frac{\partial \bar{y}_3(i\Delta)}{\partial \mathbf{c}_m} \\ \frac{\partial \bar{y}_2(i\Delta)}{\partial \mathbf{c}_m} \\ \frac{\partial \bar{y}_2(i\Delta)}{\partial \mathbf{c}_m} \\ \frac{\partial \bar{y}_2(i\Delta)}{\partial \mathbf{c}_m} \\ \frac{\partial \bar{y}_4(i\Delta)}{\partial \mathbf{c}_m} \end{bmatrix}.$$

Fig. 5.4.2 shows the mean and standard error of inferring the reaction

rates using the proposed estimator, shown as a function of the variance of the observation noise (discretization time $\Delta = 0.1$, the number of particles $N_s = 10000$, the noise variance $\sigma^2 = 1$). Several of the parameters are estimated very accurately (e.g., c_1, c_4), while others have relatively large mean-square-error (e.g., c_2, c_6). Fig. 5.5.1 compares the estimation mean-square error with the corresponding CRLB, plotted as a function of the number of measurements N used for the estimation. As indicated in Fig. 5.5.1, the estimator performs close to the CRLB for several of the parameters (e.g., c_2, c_3), while for other parameters there is room for improvement.

5.6 Conclusions

In this chapter, we studied the problem of estimating reaction rates in a gene regulatory network modeled by a chemical Langevin equation, i.e., a high-dimensional stochastic differential equation. We proposed a solution which employs a particle filtering algorithm with Markov Chain Monte Carlo move step. Extensive simulation studies demonstrated that the proposed technique requires less computational complexity to achieve performance comparable to previously proposed methods. Moreover, we considered the deterministic description of the average network dynamics based on an ordinary differential equation model. For this scenario, we computed an approximate Cramer-Rao lower bound on the mean-square error of the estimation and demonstrated that, for some of the parameters, the proposed particle filter can be nearly optimal. The computed CRLB is indicative of the number of data points (i.e.,

the number of experiments) required to achieve a desired accuracy of inferring reaction rates. Further studies are needed to enable near-CRLB performance in the scenario of estimating a large number of unknown parameters.

Chapter 6

Summary and conclusions

6.1 Summary of main results

The work presented in this thesis was motivated by the advancements of next-generation sequencing technologies and their related applications. These technologies enabled inexpensive and rapid generation of vast amounts of sequencing data. At the same time, high-throughput sequencing technologies present us with the challenge of processing and analyzing large data sets that they provide. We considered various problems in the data processing pipeline – base-calling, sequence assembly, and error correction – and developed new algorithms improving the efficiency and accuracy of data analysis. The main contributions of this thesis is summarized as follows.

- In Chapter 2, we considered the base-calling problem in Illumina's sequencing platform. We presented a particle filtering (i.e., sequential Monte Carlo) base-calling algorithm that we referred to as ParticleCall. It is based on a Hidden Markov Model (HMM) representation of the signal acquired by Illumina's sequencing-by-synthesis platforms. We also developed an new EM (PFEM) algorithm relying on the samples generated from particle filters to improve the speed of parameter estimation

in base-calling.

- In Chapter 3, we focused on the reference-guided DNA sequence assembly problem. We formulated this problem as the inference of the genome sequence on a bipartite graph and solved it using a message-passing algorithm. We showed that the proposed algorithm can be interpreted as the classical belief propagation under a certain prior. A binary reformulation of the problem led to an alternative solution in the form of another message passing algorithm that is closely related to the power iteration method. The performance of the message passing assembly algorithm is close to a genie-aided maximum a posteriori (MAP) sequence assembly scheme which is an idealized assembler with perfect quality score information and error-free mapping of the reads to their locations.

- In Chapter 4, we turned to the error correction problem in *de novo* sequence assembly. We presented a new error correction algorithm utilizing the base quality information provided by the next-generation DNA sequencing platform. The algorithm uses suffix array data structure to efficiently identify the repetitive regions between the short reads and correct the erroneous bases by comparing them with the bases in other reads. We developed a hypothesis testing scheme to improve the detection accuracy. Experimental results showed that the proposed error correction algorithm has higher accuracy compared to traditional algorithms.

- In Chapter 5, we considered an application in gene regulatory networks. We studied the problem of estimating stochastic rate constants therein. We approximated a chemical master equation description of gene regulatory networks by a related chemical Langevin equation and developed a particle filter algorithm with the Markov Chain Monte Carlo move step to solve the rate estimation problem. The algorithm can also be applied to ordinary differential equation models, which captures dynamic behavior of gene regulatory networks averaged over a large number of cells. For these models, we computed an approximation to the Cramer-Rao lower bound on the mean-square error of estimating reaction rates and demonstrated that, when the number of unknown parameters is small, the proposed particle filter can be nearly optimal.

6.2 Discussion and future directions

In the HMM we used in ParticleCall, it is possible to incorporate prior information on the transition probabilities between successive DNA bases, i.e., the transition distribution over the successive rows of S. In practice, we often do not have prior information about the transition probabilities, and hence our scheme assumed that they are uniform (i.e., bases are independent). However, it is straightforward to incorporate different transition matrix into our algorithm with other settings unaltered. Further study using other data sets is needed to examine the effect of incorporating prior information in this algorithm.

In Chapter 4, we used the histograms from the benchmark data set to approximate the conditional pdf of the test statistic S given the hypotheses. It is possible to assume these conditional distributions are subject to certain type of analytical forms such as Gaussian distribution and binomial distributions. Under these assumptions, an analytical form of the density functions can be calculated and the choice of test threshold θ can be formalized as an analytical calculation. However, whether this assumption is valid needs to be further investigated using different data sets obtained from different sequencing platforms. The process to obtain the analytical form of the conditional distribution can be further discussed.

In conclusion, we developed several Bayesian inference algorithms and showed that they improve the efficiency and accuracy of data analysis in next-generation DNA sequencing tasks. Applying such techniques to other sequencing related problems such as RNA sequencing, genotyping, and problems in metagenomics, are of great interest for future work.

Index

Bibliography

[1] J. Shendure, H. Ji, "Next-generation DNA sequencing," *Nat Biotechnology*, vol. 26, pp. 1135-1145, 2008.

[2] M. Metzker, "Emerging technologies in DNA sequencing," *Genome Research*, vol. 56, pp. 1767-1776, 2005.

[3] D. Bentley, "Whole-genome re-sequencing," *Curr Opin Genet Dev*, vol. 16, pp. 545-552, 2006.

[4] R. Nielsen, J. Paul, A. Alvrechtsen and Y. Song, "Genotype and SNP calling from next-generation sequencing data," *Nature Reviews*, vol. 12, pp. 443-451, 2011.

[5] H. Li, R. Durbin, "Fast and accurate short read alignment with Burrows-Wheeler transform," *Bioinformatics*, vol. 25, pp. 1754-1760, 2009.

[6] B. Langmead et al. "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome," *Genome Biology*, vol. 10, 2009.

[7] J. Butler, I. MacCallum, M. Kleber, I. Shlyakhter, M. Belmonte, E. Lander, C. Nusbaum and D. Jaffe, "ALLPATHS: de novo assembly of whole-genome shotgun microreads," *Genome Research*, vol. 18, no. 5, pp. 810-820, 2008.

[8] D. Zerbino and E. Birney, "Velvet: algorithms for de novo short read assembly using de Bruijn graphs," *Genome Research*, vol. 18, no. 5, pp. 810-820, 2008.

[9] M. Chaisson et al., "De novo fragment assembly with short mate-paired reads: Does the read length matter" *Genome Research*, vol. 19, pp. 336-346, 2009.

[10] J. Chen and S. Skiena, "Assembly for double -ended short-read sequencing technologies," *Advances in Genome Sequencing Technology and Algorithms*, Artech House Publishers, pp. 123-141.

[11] X. Shen and H. Vikalo, "A sequential Monte Carlo base-calling method for next-generation DNA sequencing," *2011 IEEE International Workshop on Genomic Signal Processing and Statistics (GENSIPS)*, pp. 121-122, 2011.

[12] X. Shen and H. Vikalo, "ParticleCall: A particle filter for base calling in next-generation sequencing systems," *BMC Bioinformatics*, vol. 13, July 2012.

[13] X. Shen and H. Vikalo, "A message passing algorithm for reference-guided sequence assembly from high-throughput sequencing data," *2012 IEEE International Workshop on Genomic Signal Processing and Statistics (GENSIPS)*, pp. 35-37, 2012.

[14] X. Shen, M. Shamaiah and H. Vikalo, "Message passing algorithm for inferring consensus sequence from next-generation sequencing data," *2013*

IEEE International Symposium on Information Theory, pp. 1631-1634, 2013.

[15] X. Shen, M. Shamaiah and H. Vikalo, "Iterative Learning for Reference-Guided DNA Sequence Assembly from Short Reads: Algorithms and Limits of Performance," *arXiv preprint arXiv:1403.5686*, 2014.

[16] X. Shen and H. Vikalo, "Inferring parameters of gene regulatory networks via particle filtering," *EURASIP Journal on Advances in Signal Processing*, pp. 5:1-5:9, 2010.

[17] X. Cai and X. Wang, "Stochastic modeling and simulation of gene networks," *IEEE Signal Processing Magazine*, vol. 24, no. 1, pp. 27-36, 2007.

[18] M. Fedurco, A. Romieu, S. Williams and et al., "BTA, a novel reagent for DNA attachment on glass and efficient generation of solid-phase amplified DNA colonies," *Nucleic Acids Res*, vol. 34, no. 3, 2006.

[19] G. Turcatti and A. Romieu and M. Fedurce and et al., "A new class of cleavable fluorescent nucleotides: synthesis and optimization as reversible terminators for DNA sequencing by synthesis," *Nucleic Acids Res*, vol. 36, no. 4, 2008.

[20] C. Ledergerber and C. Dessimoz, "Base-calling for next-generation sequencing platforms," *Briefings in Bioinformatics*, vol. 12, pp. 489-497, 2011.

[21] J. Rougemont, A. Amzallag, C. Iseli, L. Farinelli, I. Xenarios and F. Naef, "Probabilistic base calling of solexa sequencing data," *BMC Bioinformatics*, vol. 9:431, 2008.

[22] Y. Erlich, P. Mitra, M. Delabastide, W. McCombie and G. Hannon, "AltaCyclic: a self-optimizing base caller for next-generation sequencing," *Nat Methods*, vol. 5, pp. 679-682, 2008.

[23] W. Kao, K. Stevens and Y. Song, "BayesCall: A model-based base-calling algorithm for high-throughput short-read sequencing," *Genome Research*, vol. 19, pp. 1884-1895, 2009.

[24] W. Kao, K. Stevens and Y. Song, "naiveBayesCall: an efficient model-based base-calling algorithm for high-throughput sequencing," *Journal of Computational Biology*, vol. 18, pp. 365-377, 2011.

[25] O. Cappé, E. Moulines and T. Rydén, "Inference in hidden Markov models," Springer Verlag, 2005.

[26] M. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters," *Journal of the American Statistical Association*, pp. 590-599, 1999.

[27] A. Doucet, S. Godsill and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statistics and computing*, vol. 10, pp. 197-208, 2000.

[28] S. Kim, N. Shephard and S. Chib, "Stochastic volatility: likelihood inference and comparison with ARCH models," *The Review of Economic Studies*, vol. 65, no. 3, pp. 361-393, 1998.

[29] M. Shamaiah, X. Shen and H. Vikalo, "Sequential Monte Carlo method for parameter estimation in diffusion models of affinity-based biosensors," *2011 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 525-528, 2011.

[30] M. Shamaiah, X. Shen and H. Vikalo, "Estimating Parameters of Sampled Diffusion Processes in Affinity Biosensors," *IEEE Transactions on Signal Processing*, vol. 60, pp. 3228-3239, 2012.

[31] J. Liu and R. Chen, "Sequential Monte Carlo methods for dynamic systems," *Journal of the American statistical association*, pp. 1032-1044, 1998.

[32] G. Kitagawa, "Monte Carlo filter and smoother for non-Gaussian nonlinear state space models," *Journal of computational and graphical statistics*, pp. 1-25, 1996.

[33] J. Carpenter, P. Clifford and P. Fearnhead, "Improved particle filter for nonlinear problems," *Radar, Sonar and Navigation, IEE Proceedings-*, vol. 146, no. 1, pp. 2-7, 1999.

[34] A. Doucet and X. Wang, "Monte Carlo methods for signal processing: A review in the statistical signal processing context," *IEEE Signal Processing*

Magzine, vol. 22, pp. 152-170, 2005.

[35] S. Eddy, "Hidden Markov models," *Current Opinion in Structural Biology*, vol. 6, no. 3, pp. 361-365, 1996.

[36] A. Aktmann, P. Weber, et al. "A beginners guide to SNP calling from high-throughput DNA-sequencing data," *Human Genetics*, pp. 1-14, 2012.

[37] M. Shamaiah, "Algorithms and analysis for next generation biosensing and sequencing systems," *PhD Thesis*, 2012.

[38] R. Dalloul, et al. "Multi-platform next-generation sequencing of the domestic turkey (Meleagris gallopavo): Genome assembly and analysis," *PLoS Biol*, 8:e1000475, 2010.

[39] R. Li, et al. "The sequence and de novo assembly of the giant panda genome," *Nature*, vol. 463, pp. 311-317, 2010.

[40] R. Li, Y. Li, et al. "SNP detection for massively parallel whole-genome resequencing," *Genome Research*, vol. 19, pp. 1124-1132, 2009.

[41] H. Li, B. Handsaker, et al. "The Sequence Alignment/Map format and SAMtools," *Bioinformatics*, vol. 25, pp. 2078-2079, 2009.

[42] M. DePristo, E. Banks, R. Poplin, et al. "A framework for variation discovery and genotyping using next-generation DNA sequencing data," *Nature Genetics*, vol. 43, pp. 491-498, 2011.

[43] D. Karger, S. Oh, D. Shah, "Iterative learning for reliable crowd-sourcing systems," *in Proceedings of NIPS*, 2011.

[44] D. Karger, S. Oh, D. Shah, "Budget-optimal Crowdsourcing using Low-rank Matrix Approximation," *Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on. IEEE*, 2011.

[45] J. Yedidia, W. Freeman, Y. Weiss, "Constructing free-energy approximations and generalized belief propagation algorithms," *IEEE Transactions on Information Theory*, vol 51, pp. 2282-2312, 2005.

[46] R. M. Durbin et al. "A map of human genome variation from population-scale sequencing," *Nature* 467 (7319), pp. 1061-1073, 2010.

[47] F. Kschischang, H. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, 2001.

[48] X. Lin, S. Yacoub, J. Burns, and S. Simske, "Performance analysis of pattern classifier combination by plurality voting," *Pattern Recognition Letters*, vol. 24, pp. 1959-1969, 2002.

[49] W. C. Kao, A. H. Chan, and Y. S. Song, "ECHO: a reference-free short-read error correction algorithm," *Genome Research*, vol. 21, no. 7, pp. 1181-92, 2011.

[50] G. Levitin, *Universal Generating Function in Reliability Analysis and Optimization*, Springer-Verlag, 2005.

[51] G. A. Churchill and M. S. Waterman, "The accuracy of DNA sequences: Estimating sequence quality," *Genomics*, vol. 14, pp. 89-98, 1992.

[52] DNASTAR: http://www.dnastar.com/t-sub-nextgen-genome-solutions-automated-genome-closure.aspx

[53] T. Rausch, K. Sergey, et al. "A consistency-based consensus algorithm for de novo and reference-guided sequence assembly of short reads," *Bioinformatics*, vol. 25, no. 9, pp. 1118-1124, 2009.

[54] J. Schröder, H. Schröder, S. Puglisi, R. Sinha and B. Schmidt, "SHREC: a short-read error correction method," *Bioinformatics*, vol. 25, no. 17, pp. 2157-2163, 2009.

[55] L. Ilie, F. Fazayeli and S. Ilie, "HiTEC: accurate error correction in high-throughput sequencing data," *Bioinformatics*, vol. 27, no. 3, pp. 295-302, 2011.

[56] U. Manber and G. Myers, "Suffix arrays: a new method for on-line search," *SIAM J. Comput.*, 2003.

[57] J. Karkkainen and P. Sanders, "Simple linear work suffix array construction," *in Proceedings of ICALP'03*, 2003.

[58] D.K. Kim et al., "Constructing suffix arrays in linear time," *J. Discrete Algorithms*, vol. 3, pp. 126-142, 2005.

[59] T. Kasai et al., "Linear-time longest-common-prefix computation in suffix arrays and its applications," *in Proceedings of CPM'01*, 2001.

[60] Y. Mori,"libdivsufsort," *http://code.google.com/p/libdivsufsort/*.

[61] W. F. Loomis and P. W. Sternberg, "Genetic networks," *Science*, pp: 269-649, 1995.

[62] D. Thieffry, "From global expression data to gene networks," *BioEssays*, vol. 21, no. 11, pp: 895-899, 1999.

[63] R. Albert, "Boolean modeling of genetic regulatory networks," *Complex Networks*, Springer-Verlag, 2004.

[64] D. T. Gillespie, "Exact stochastic simulation of coupled chemical reaction," *Journal of Physical Chemistry*, vol. 81, no. 25, pp. 2340-2361, 1977.

[65] D. T. Gillespie, "The chemical Langevin equation," *Journal of Chemical Physics*, vol. 113, no. 1, pp. 297-306, 2000.

[66] D. T. Gillespie, "A rigorous derivation of the chemical master equation," *Physica A*, 188: 404-425, 1992.

[67] H. H. McAdams and A. Arkin, "Stochastic mechanisms in gene expression," *Proceedings of the National Academy of Sciences*, vol. 94, pp. 814-819, 1997.

[68] T. Chen, H. L. He, and G. M. Church, "Modeling gene expression with differential equations," *Proceedings of Pacific Symposium on Biocomputing*, pp. 29-40, 1999.

[69] F. Grognard, H. de Jong, and J.-L. Gouze, "Piecewise-linear models of genetic regulatory networks: theory and examples," *Lecture notes in control and information sciences (LNCIS)*, Springer-Verlag, 2007.

[70] N. Friedman, M. Linial, I. Nachman, and D. Pe'er, "Using Bayesian networks to analyze expression data," *J. Comput. Biol.*, vol. 7, 601, 2000.

[71] D. Heckerman, "A tutorial on learning with Bayesian networks," in *Learning in Graphical Models*, Kluwer, 1998.

[72] S. A. Kauffman, "Metabolic stability and epigenesis in randomly constructed genetic nets," *J. of Theoretical Biology*, vol. 22, pp: 437-467, 1969.

[73] R. J. Boys, D. J. Wilkinson, and T. B. L. Kirkwood, "Bayesian inference for a discretely observed stochastic kinetic model," *Statistics and Computing*, vol. 18, no. 2, pp. 125-135, 2008.

[74] K.-C. Chen, et. al., "A stochastic differential equation model for quantifying transcriptional regulatory network in Saccharomyces cerevisiae," *Bioinformatics*, vol. 21, no. 12, pp: 2883-2890, 2005.

[75] J. Berg, "Dynamics of gene expression and the regulatory inference problem" *Europhys. Lett.*, vol. 82, 28010, 2008.

[76] A. Benecke, "Gene regulatory network inference using out of equilibrium statistical mechanics," *HFSP Journal*, vol. 2, no. 4, pp: 183-8, 2008.

[77] A. Golightly and D. J. Wilkinson, "Bayesian sequential inference for stochastic kinetic biochemical network models," *Journal of Computational Biology*, 13(3), 838-851, 2006.

[78] A. Golightly, "Bayesian Inference for Nonlinear Multivariate Diffusion Processes," *Ph.D Thesis*, Newcastle University, 2006.

[79] H. Cramer, "Mathematical Models of Statistics", Princeton University Press, Princeton, NJ 1946.

[80] Z. Li and M. Osborne, "Parameter estimation of ordinary differential equations," *IMA Journal of Numerical Analysis*, 25, 264-285, 2005.

[81] R. Srivastava, L. You, J. Summers, J. Yin, "Stochastic vs. Deterministic Modeling of Intracellular Viral Kinetics," *Journal of Theoretical Biology*, 218, 2002.

[82] H. Vikalo, B. Hassibi, A. Hassibi, "Limits of Performance of Quantitative Polymerase Chain Reaction Systems", *IEEE Tran. on Information Theory*, Vol.56, No.2, 2010.

[83] J. Goutsias, "Quasiequilibrium approximation of fast reaction kinetics in stochastic biochemical systems," *Journal of Chemical Physics*, 122, 2005.