BMC Bioinformatics

METHODOLOGY ARTICLE Open Access

ParticleCall: A particle filter for base calling in next-generation sequencing systems

Xiaohu Shen and Haris Vikalo*

Abstract

Background: 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. Accuracy and lengths of their reads, however, are yet to surpass those provided by the conventional Sanger sequencing method. This motivates the search for computationally efficient algorithms capable of reliable and accurate detection of the order of nucleotides in short DNA fragments from the acquired data.

Results: In this paper, we consider Illumina's sequencing-by-synthesis platform which relies on reversible terminator chemistry and describe the acquired signal by reformulating its mathematical model as 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 a data set obtained by sequencing phiX174 bacteriophage using Illumina's Genome Analyzer II. The results show that the developed base calling scheme is significantly more computationally efficient than the best performing unsupervised method currently available, while achieving the same accuracy.

Conclusions: The proposed ParticleCall provides more accurate calls than the Illumina's base calling algorithm, Bustard. At the same time, ParticleCall is significantly more computationally efficient than other recent schemes with similar performance, rendering it more feasible for high-throughput sequencing data analysis. Improvement of base calling accuracy will have immediate beneficial effects on the performance of downstream applications such as SNP and genotype calling.
ParticleCall is freely available at https://sourceforge.net/projects/particlecall.

Background

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. 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.

A widely used sequencing-by-synthesis platform, commercialized by Illumina, relies on reversible terminator chemistry. 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 [5]. Such schemes include work presented in [6-9]. Among the proposed methods, the BayesCall algorithm [8] 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

*Correspondence: hvikalo@ece.utexas.edu
Department of Electrical and Computer Engineering, University of Texas at Austin, 1 University Station C0803, Austin, TX, 78712-0240, US

in downstream applications. However, its performance gains come at high computational costs. A modified version of the BaseCall algorithm named naiveBayesCall [9] 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 paper, we propose 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. 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. Finally, we demonstrate that ParticleCall has the best discrimination ability among all of the considered base calling schemes.

Methods

In this section, we first review the data acquisition process and the basic mathematical model of the Illumina's sequencing-by-synthesis platform. Then 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.

Illumina sequencing platform

A sequencing task on the Illumina's 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 [10]) and subjected to bridge amplification in order to create a cluster of identical copies of DNA templates [11]. 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 CCD 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.

Overview of the mathematical model

To describe the signal acquired by the Illumina's sequencing-by-synthesis platform, a parametric model was proposed in [8]. 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,

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

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

$$\mathbf{x}_t = \lambda_t S \mathbf{h}_t, \tag{2}$$

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

$$\mathbf{y}_t = K_t \mathbf{x}_t + \sum_{b \in \{A,C,G,T\}} x_t^b \eta_t^b, \tag{3}$$

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) and (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, $H_{t-l,t}, H_{t-l+1,t}, \ldots, H_{t,t}, \ldots H_{t+r-1,t}, H_{t+r,t}$, and then expression (2) becomes

$$\mathbf{x}_t \approx \mathbf{x}_t^w = \lambda_t S \mathbf{h}_t^w = \lambda_t \sum_{i=-l}^{r} \mathbf{s}_{t+i} H_{t+i,t}. \tag{4}$$

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

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

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

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) [12]. 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

$$\mathbf{y}_t' | S, \lambda_t \sim \mathcal{N}\left(K_t \mathbf{x}_t^w, \|\mathbf{x}_t^w\|_2^2 \Sigma_t\right). \tag{5}$$

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.,

$$S_t^w = [\, \mathbf{s}_{t-l} \; \mathbf{s}_{t-l+1} \; \ldots \; \mathbf{s}_t \; \ldots \; \mathbf{s}_{t+r} \,]. \tag{6}$$

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 Figure 1. The observation dynamics that characterize the

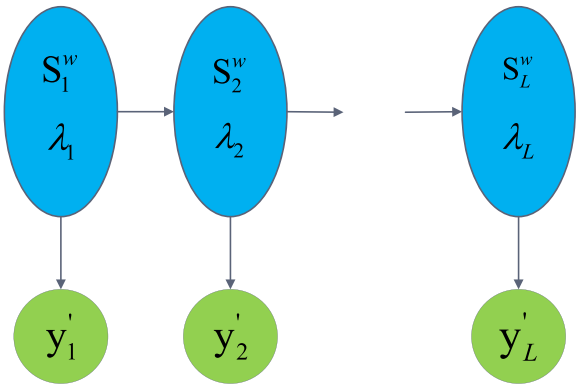

Figure 1 A hidden Markov model of the generated signal in Illumina sequencing-by-synthesis platforms. An illustration of the graphical HMM of the Illumina's sequencing platform. The observations $\mathbf{y}'$ represent signal intensities after the removal of residual effects. The states are the combinations of S_t^w and λ_t, which represent a subsequence of the template centered at position t and per-cluster density, respectively.

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 (5) that

$$g\left(\mathbf{y}_t^{'}|S_t^w, \lambda_t\right) \sim \mathcal{N}\left(K_t\mathbf{x}_t^w, \|\mathbf{x}_t^w\|_2^2\Sigma_t\right). \quad (7)$$

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

$$f(S_t^w, \lambda_t|S_{t-1}^w, \lambda_{t-1}) = f_1(S_t^w|S_{t-1}^w)f_2(\lambda_t|\lambda_{t-1}). \quad (8)$$

The second term on the right-hand side of (8), $f_2(\lambda_t|\lambda_{t-1})$, is known from the density decay model (1),

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

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

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

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

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

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.

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 (7)-(8) 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 [13]. 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 [14] to calculate $p(S_t^w, \lambda_t|y'_{1:t})$. Different choices and approximation methods of proposal densities are considered in [15-17]. We directly use the transition (8) 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 [14], residual resampling [18] and systematic resampling [19,20]. 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 and associated weights by w, 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 [14]. 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 $i = 1 \rightarrow N_p$ **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 $w_1^{(i)} \propto g(\mathbf{y}'_1|S_1^{w,(i)}, \lambda_1^{(i)})$ as in (7).
2. Run iteration $t(t \geq 2)$:
2.1 Sampling:
for $i = 1 \rightarrow N_p$ **do**
Sample $S_t^{w,(i)}, \lambda_t^{(i)} \sim f(\cdot,\cdot|S_{t-1}^{w,(i)}, \lambda_{t-1}^{(i)})$ according to (8).
end for
2.2 Update the importance weight

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

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 (8), 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 [16]. 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

$$p\left(\lambda_{1:t}, S_{1:t}^w|\mathbf{y}'_{1:t}\right) = p\left(S_{1:t}^w|\mathbf{y}'_{1:t}, \lambda_{1:t}\right) p\left(\lambda_{1:t}|\mathbf{y}'_{1:t}\right).$$

Since $p(\lambda_{1:t}|\mathbf{y}'_{1:t}) \propto p(\mathbf{y}'_t|\mathbf{y}'_{1:t-1}, \lambda_{1:t}) p(\lambda_t|\lambda_{t-1}^{(i)})$, where $\lambda_{t-1}^{(i)}$ 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 $i = 1 \rightarrow N_p$ **do**
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 $w_1^{(i)} \propto g(\mathbf{y}'_1|\lambda_1^{(i)}) \propto \sum_{S_1^w} g(\mathbf{y}'_1|S_1^w, \lambda_1^{(i)})$.
1.3 Calculate the discrete distribution $p(S_1^w|\mathbf{y}_1, \lambda_1^{(i)})$ for each i.
2. Run iteration $t(t \geq 2)$:
2.1 Sampling:
for $i = 1 \rightarrow N_p$ **do**
Sample $\lambda_t^{(i)} \sim f(\cdot|\lambda_{t-1}^{(i)})$.
end for
2.2 Update the importance weight $w_t^{(i)} \propto w_{t-1}^{(i)} g(\mathbf{y}'_t|\mathbf{y}'_{1:t-1}, \lambda_{1:t}^{(i)})$. and normalize the weights.
2.3 Resample if $K_{\text{eff}} \leq N_{threshold}$
2.4 Update $p(S_t^w|\mathbf{y}'_{1:t}, \lambda_{1:t}^{(i)})$
for $i = 1 \rightarrow N_p$ **do**
Update $p(S_t^w|\mathbf{y}'_{1:t}, \lambda_{1:t}^{(i)})$ using $p(S_{t-1}^w|\mathbf{y}'_{1:t-1}, \lambda_{1:t-1}^{(i)})$ and $\lambda_t^{(i)}$.
end for

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

$$\begin{aligned} & g\left(\mathbf{y}'_t|\mathbf{y}'_{1:t-1}, \lambda_{1:t}^{(i)}\right) \\ &= \sum_{S_t^w} p\left(\mathbf{y}'_t|\mathbf{y}'_{1:t-1}, S_t^w, \lambda_{1:t}^{(i)}\right) p\left(S_t^w|\mathbf{y}'_{1:t-1}, \lambda_{1:t}^{(i)}\right) \\ &= \sum_{S_t^w} p\left(\mathbf{y}'_t|S_t^w, \lambda_t^{(i)}\right) \sum_{S_{t-1}^w} \Big[p\left(S_t^w|S_{t-1}^w, \mathbf{y}'_{1:t-1}, \lambda_{1:t}^{(i)}\right) \\ &\qquad \times p\left(S_{t-1}^w|\mathbf{y}'_{1:t-1}, \lambda_{1:t}^{(i)}\right)\Big], \end{aligned} \quad (9)$$

where $p(\mathbf{y}'_t|S_t^w, \lambda_t^{(i)})$ is the observation density, $p(S_{t-1}^w|\mathbf{y}'_{1:t-1}, \lambda_{1:t}^{(i)}) = p(S_{t-1}^w|\mathbf{y}'_{1:t-1}, \lambda_{1:t-1}^{(i)})$ due to the independence of the state transitions, and $p(S_t^w|S_{t-1}^w, \mathbf{y}'_{1:t-1}, \lambda_{1:t}^{(i)}) =$

$p(S_t^w|S_{t-1}^w)$ 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

$$\begin{aligned} p\left(S_t^w|\mathbf{y}'_{1:t},\lambda_{1:t}^{(i)}\right) &\propto p\left(S_t^w,\mathbf{y}'_t,\lambda_t^{(i)}|\mathbf{y}'_{1:t-1},\lambda_{1:t-1}^{(i)}\right)\\ &=\sum_{S_{t-1}^w}\Big[p\left(\mathbf{y}'_t,S_t^w,\lambda_t^{(i)}|\mathbf{y}'_{1:t-1},S_{t-1}^w,\lambda_{1:t-1}^{(i)}\right)\\ &\qquad\times p\left(S_{t-1}^w|\mathbf{y}'_{1:t-1},\lambda_{1:t-1}^{(i)}\right)\Big]\\ &=\sum_{S_{t-1}^w}\Big[p\left(\mathbf{y}'_t|S_t^w,\lambda_t^{(i)},\mathbf{y}'_{1:t-1},S_{t-1}^w,\lambda_{1:t-1}^{(i)}\right)\\ &\quad\times p\left(S_t^w,\lambda_t^{(i)}|\mathbf{y}'_{1:t-1},S_{t-1}^w,\lambda_{1:t-1}^{(i)}\right)\\ &\quad\times p\left(S_{t-1}^w|\mathbf{y}'_{1:t-1},\lambda_{1:t-1}^{(i)}\right)\Big]\\ &=p\left(\mathbf{y}'_t|S_t^w,\lambda_t^{(i)}\right)p\left(\lambda_t^{(i)}|\lambda_{t-1}^{(i)}\right)\\ &\quad\times\sum_{S_{t-1}^w}p\left(S_t^w|S_{t-1}^w\right)p\left(S_{t-1}^w|\mathbf{y}'_{1:t-1},\lambda_{1:t-1}^{(i)}\right) \end{aligned} \tag{10}$$

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 [8]. In section Results and discussion, 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.

Assumptions on parameters

Recall that the set of parameters needed to run 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., [8]). 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 [5]. 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.

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

$$\mathbf{x}_t = \lambda_t \mathbf{s}_t. \tag{11}$$

Replacing (2) by (11) 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

$$\Theta_1^i = \arg\max_{\Theta_1} L_1\left(\Theta_1^{i-1}\right), \tag{12}$$

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

$$\begin{aligned} L_{1,j}(\Theta_1^{i-1}) &= \log P(\mathbf{y}_{1:w}|\Theta_1^{i-1})\\ &= \mathbf{E}\left[\log P(\mathbf{y}_{1:w},\mathbf{s}_{1:w},\lambda_{1:w}|\Theta_1^{i-1})\right], \end{aligned} \tag{13}$$

Table 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

ParticleCall is run using parameters obtained via the MCEM parameter estimation scheme as well as via the PFEM parameter estimation algorithm proposed in this paper. Rao-Blackwellized ParticleCall is run using parameters via the MCEM parameter estimation scheme.

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 (7) and (8), 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 (13). The maximization (12) 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.

Results and discussion

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.

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 paper, we set $l = r = 1$. The accuracy of base-calling for different N_p is shown in Table 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 (9) and (10) 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 shows the ParticleCall base calling error rate and parameter estimation times for different window lengths w. In the remainder of the paper, we set $w = 5$ as it

Table 2 ParticleCall parameter estimation

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

ParticleCall base-calling error rate and the parameter estimation time of the proposed PFEM parameter estimation algorithm.

Table 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

The base-calling error rate and the running times of different algorithms. ParticleCall is run using parameters obtained via the MCEM parameter estimation scheme as well as via the PFEM parameter estimation algorithm proposed in this paper. For Bustard and Rolexa, only the total running times are reported.

leads to desirable performance/speed characteristics of the algorithm.

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 paper. The results are reported in Table 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 Figure 2. It can be seen from Table 3 and Figure 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. Figure 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 [5]. Moreover, we see from Table 3 that performing parameter estimation via the MCEM algorithm proposed in [8] requires 19 hours, while the particle filter implementation of the EM estimation scheme proposed in this paper takes only 39 minutes. As evident from Table 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,

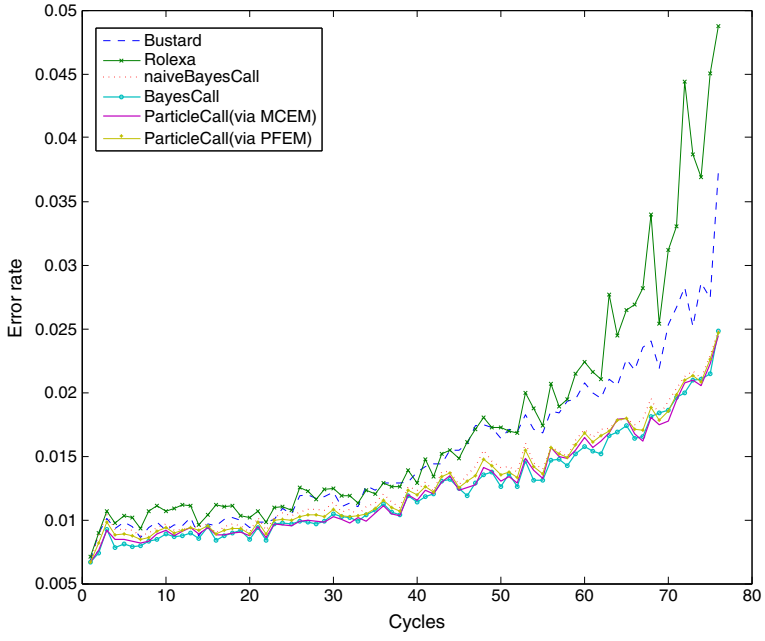

Figure 2 Per-cycle error rates of ParticleCall, BayesCall, naiveBayesCall, Rolexa and Bustard. The figure compares the per-cycle error rates of different base-calling algorithms. ParticleCall and BayesCall are the most accurate ones.

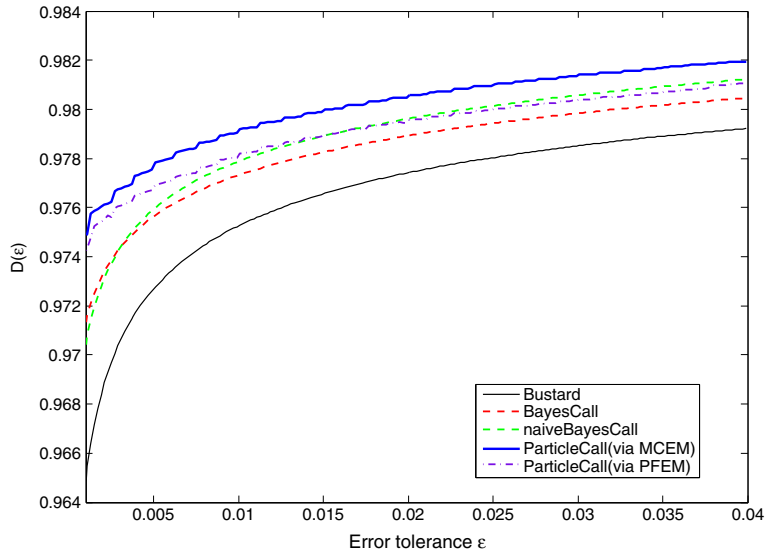

Figure 3 Discrimination ability $D(\epsilon)$ of quality scores vs error tolerance. The figure shows the percentage of correctly called bases under different error tolerance ϵ.

which is 9 times faster than the total time required by the less accurate naiveBayesCall.

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

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

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 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. Figure 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

Table 4 de novo assembly results

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

The maximum contig length and N50 length of *de novo* assembly using Velvet. The average values over 200 experiments are shown in the table.

quality scores below ϵ are considered invalid, ParticleCall provides the most accurate results among the considered base-calling schemes.

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 [21,22]. 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 [22] 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 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.

Conclusions

In this paper we presented 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.

Author's contributions

Algorithms and experiments were designed by Xiaohu Shen (XS) and Haris Vikalo (HV). Algorithm code was implemented and tested by XS. The manuscript was written by XS and HV. Both authors read and approved the final manuscript.

Acknowledgements

This work was funded by the National Institute of Health under grant 1R21HG006171-01.

Competing interests

The authors declare that they have no competing interests.

Received: 14 March 2012 Accepted: 9 July 2012
Published: 9 July 2012

References

1. Shendure J, Ji H: **Next-generation DNA sequencing.** *Nat Biotechnology* 2008, **26:**1135–1145.
2. Metzker M: **Emerging technologies in DNA sequencing.** *Genome Research* 2005, **56:**1767–1776.
3. Bentley D: **Whole-genome re-sequencing.** *Curr Opin Genet Dev* 2006, **16:**545–552.
4. Nielsen R, Paul JS, Alvrechtsen A, Song YS: **Genotype and SNP calling from next-generation sequencing data.** *Nature Reviews* 2011, **12:**443–451.
5. Ledergerber C, Dessimoz C: **Base-calling for next-generation sequencing platforms.** *Briefings in Bioinformatics* 2011, **12:**489–497.
6. Rougemont J, Amzallag A, Iseli C, Farinelli L, Xenarios I, Naef F: **Probabilistic base calling of solexa sequencing data.** *BMC Bioinformatics* 2008, **9:**431.
7. Erlich Y, Mitra P, Delabastide M, McCombie W, Hannon G: **Alta-Cyclic: a self-optimizing base caller for next-generation sequencing.** *Nat Methods* 2008, **5:**679–682.
8. Kao W, Stevens K, Song Y: **BayesCall: A model-based base-calling algorithm for high-throughput short-read sequencing.** *Genome Research* 2009, **19:**1884–1895.
9. Kao W, Stevens K, Song Y: **naiveBayesCall: an efficient model-based base-calling algorithm for high-throughput sequencing.** *Journal of Computational Biology* 2011, **18:**365–377.
10. Fedurco M, Romieu A, Williams S, *et al*: **BTA, a novel reagent for DNA attachment on glass and efficient generation of solid-phase amplified DNA colonies.** *Nucleic Acids Res* 2006, **34**(3):e22.
11. Turcatti G, Romieu A, Fedurce M, *et al*: **A new class of cleavable fluorescent nucleotides: synthesis and optimization as reversible terminators for DNA sequencing by synthesis.** *Nucleic Acids Res* 2008, **36**(4):e25.
12. Eddy S: **Hidden Markov models.** *Current Opinion in Structural Biology* 1996, **6**(3):361–365.
13. Doucet A, Wang X: **Monte Carlo methods for signal processing: A review in the statistical signal processing context.** *IEEE Signal Processing Magzine* 2005, **22:**152–170.
14. Cappé O, Moulines E, Rydén T: *Inference in hidden Markov models*. New York: Springer Verlag ; 2005.
15. Pitt M, Shephard N: **Filtering via simulation: Auxiliary particle filters.** *Journal of the American Statistical Association* 1999, **94:**590–599 .
16. Doucet A, Godsill S, Andrieu C: **On sequential Monte Carlo sampling methods for Bayesian filtering.** *Statistics and computing* 2000, **10**(3):197–208.
17. Kim S, Shephard N, Chib S: **Stochastic volatility: likelihood inference and comparison with ARCH models.** *The Review of Economic Studies* 1998, **65**(3):361–393.
18. Liu J, Chen R: **Sequential Monte Carlo methods for dynamic systems.** *Journal of the American statistical association* 1998, **93:**1032–1044.
19. Kitagawa G: **Monte Carlo filter and smoother for non-Gaussian nonlinear state space models.** *Journal of computational and graphical statistics* 1996, **5:**1–25.
20. Carpenter J, Clifford P, Fearnhead P: **Improved particle filter for nonlinear problems.** In *Radar, Sonar and Navigation, IEE Proceedings-, Volume 146,* IET; 1999:2–7 .
21. Butler J, MacCallum I, Kleber M, Shlyakhter I, Belmonte M, Lander E, Nusbaum C, Jaffe D: **ALLPATHS: de novo assembly of whole-genome shotgun microreads.** *Genome Research* 2008, **18**(5):810–820.
22. Zerbino D, Birney E: **Velvet: algorithms for de novo short read assembly using de Bruijn graphs.** *Genome research* 2008, **18**(5):821–829.

doi:10.1186/1471-2105-13-160
Cite this article as: Shen and Vikalo: ParticleCall: A particle filter for base calling in next-generation sequencing systems. *BMC Bioinformatics* 2012 **13**:160.