Multiple Sequence Alignment with Hidden Markov Models Learned by Random Drift Particle Swarm Optimization

Jun Sun, Vasile Palade, Xiaojun Wu, and Wei Fang

Abstract—Hidden Markov Models (HMMs) are powerful tools for multiple sequence alignment (MSA), which is known to be an NP-complete and important problem in bioinformatics. Learning HMMs is a difficult task, and many meta-heuristic methods, including particle swarm optimization (PSO), have been used for that. In this paper, a new variant of PSO, called the random drift particle swarm optimization (RDPSO) algorithm, is proposed to be used for HMM learning tasks in MSA problems. The proposed RDPSO algorithm, inspired by the free electron model in metal conductors in an external electric field, employs a novel set of evolution equations that can enhance the global search ability of the algorithm. Moreover, in order to further enhance the algorithmic performance of the RDPSO, we incorporate a diversity control method into the algorithm and, thus, propose an RDPSO with diversity-guided search (RDPSO-DGS). The performances of the RDPSO, RDPSO-DGS and other algorithms are tested and compared by learning HMMs for MSA on two well-known benchmark data sets. The experimental results show that the HMMs learned by the RDPSO and RDPSO-DGS are able to generate better alignments for the benchmark data sets than other most commonly used HMM learning methods, such as the Baum-Welch and other PSO algorithms. The performance comparison with well-known MSA programs, such as ClustalW and MAFFT, also shows that the proposed methods have advantages in multiple sequence alignment.

Index Terms—Hidden Markov Models, multiple sequence alignment, parameter learning, particle swarm optimization

1 Introduction

Multiple sequence alignment (MSA) of nucleotides, or amino acids, is one of the most important and challenging problems in bioinformatics. It is an extension of a pairwise alignment to incorporate more than two sequences at a time. Multiple alignment methods try to align all of the sequences in a given query set, and the resulting aligned sequences are often used to construct phylogenetic trees, to find protein families, to predict secondary and tertiary structures of new sequences, and to demonstrate the homology between new sequences and existing families [1]. MSAs require more sophisticated methodologies than pairwise alignments since they are more computationally complex, and most formulations of the problem lead to NP-complete combinatorial optimization problems [2].

The dynamic programming method is theoretically applicable to any number of sequences. However, since it is computationally expensive in both time and memory, it is rarely used for more than three or four sequences in its most basic form [1]. One way to tackle this problem is to use a heuristic search, known as the "progressive alignment" technique, that builds up a final alignment by combining pairwise alignments, beginning with the most similar pair and progressing to the most distantly related ones [3], [4]. The most popular MSA program using progressive alignments is ClustalW, which is suitable for alignments of sequence sets with medium lengths [3]. T-Coffee is another common progressive alignment program, which is slower than ClustalW, but generally produces more accurate alignments for distantly related sequence sets [5]. ProbCons is a program for progressive alignment of protein sequences, i.e., a progressive protein MSA tool [6]. It incorporates the so-called probabilistic consistency technique, a novel scoring function for comparing multiple sequences.

Another set of approaches for MSA, known as the iterative methods, work similarly to progressive methods, but repeatedly align the initial sequences as well as adding new sequences to the growing MSA set. They produce multiple sequence alignments while reducing the errors inherent in progressive methods. MUSCLE is a popular iterative method and it improves on the progressive methods by using a more accurate distance measure to assess the relatedness of two sequences [7]. MAFFT is another popular MSA program, which incorporates different strategies including progressive methods (PartTree, FFT-NS-1, and L-INS-1), iterative methods (FFT-NS-i, L-INS-I, and G-INS-i) and structural alignment methods (Q-INS-I and X-INS-i) [8].

An alternative to progressive and iterative alignment methods is to employ stochastic optimization methods, such as the simulated annealing (SA) [9], [10] or evolutionary algorithms (EAs) [11], [12]. These approaches optimize an objective function, which measures the quality of multiple sequence alignment by updating the candidate alignments until an optimal alignment is found.

- *J. Sun, X. Wu, and W. Fang are with the Key Laboratory of Advanced Control for Light Industry (Ministry of China), Jiangnan University, Wuxi, Jiangsu 214122, China. E-mail: {sunjun_wx, wxfangwei}@hotmail.com, wu_xiaojun@yahoo.com.cn.*
- *V. Palade is with the Faculty of Engineering and Computing, Coventry University, Priory Street, Coventry, CV1 5FB, UK. E-mail: vasile.palade@coventry.ac.uk.*

Manuscript received 24 Dec. 2012; revised 22 June 2013; accepted 4 Nov. 2013; date of publication 19 Nov. 2013; date of current version 7 May 2014. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TCBB.2013.148

PSO不是global也不是local search

Other efficient approaches are based on probabilistic models, such as the Hidden Markov Models (HMMs), which currently represent one of the most popular techniques for multiple sequence alignment [13], [14], [15], [16]. There are several alignment programs in which variants of HMM-based methods have been implemented. These programs are noted for their scalability and efficiency, although using an HMM method is more complex than using more common progressive methods. The most well-known HMM-based software packages for MSA are sequence alignment and modeling system (SAM) [17] and HMMER [18].

In order to use an HMM method for MSA, one has to perform the parameter learning, or the training task first; that is, to find the best set of state transition and output probabilities for an HMM with a given set of output sequences. The resulting HMM is then employed to create a sequence of gap insertions and deletion instructions to align the sequences. Generally, an HMM topology used for the MSA problem requires roughly as many states as the average length of the sequences in the problem. As such, one issue of the parameter learning in HMMs is that there is no known deterministic algorithm that can guarantee to yield an optimally learned HMM within reasonable computational time. The most common way to deal with this problem is to employ an approximation algorithm based on statistics and re-estimation. The Baum-Welch (BW) algorithm, known as the forward-backward algorithm, is the most widely used example of such algorithms [16]. The gradient methods [13] were also used to estimate the parameters of an HMM, but these methods are local search techniques that usually result in sub-optimally trained HMMs. Another possibility is to estimate the parameters of an HMM by random optimization algorithms, such as the SA [19] and EAs [20]. For example, the well-known alignment software HMMER employs SA for HMM training. However, there always are complains that the SA and EAs encounter some problems, such as lack of local search ability, premature convergence and slow convergence speed. Particle swarm optimization (PSO) algorithm, a relatively recent population-based random search technique, has demonstrated its better performance than EAs and SA in HMM training for MSA [21], [22].

This study focuses on HMM-based methods for MSA due to their scalability and efficiency. However, the number of states and parameters of an HMM increases with the average length of the sequences to be aligned, so that the dimensionality and complexity of the training problem of the HMM also increase with the average length. Although the HMM can easily be scaled to align large sequences efficiently, its training task is essentially a high-dimensional and multimodal optimization problem challenging to solve. Thus, our goal in this study is to develop an efficient global optimization method to train the HMM for MSA.

After extensive and in-depth study, we selected the PSO algorithm as a candidate to be modified to achieve our goal of training HMMs for MSA. The reason why PSO was so attractive to us was that it has many advantages, such as faster convergence speed, lower computational requirements, and it can be easily parallelized and has fewer parameters to adjust. However, PSO has the following shortcomings. First of all, it has been theoretically proved that the PSO algorithm is not a global search algorithm, even not a local one, according to the convergence criteria [23]. Practically, the algorithm is more prone to encounter premature convergence for high-dimensional problems, due to the weakening of its global search ability during the middle and later stages of the search process. Additionally, PSO is widely known to be sensitive to its search parameters, and even to the so called "swarm topology", so that users may feel awkward when selecting the parameters and the topology when using the algorithm [24]. Finally, the performance of PSO appears to be very sensitive to the setting of up-bound and lower-bound of the search scope [24]. If the global solution is located near the boundary of the search scope, the algorithm may have little chance to catch it. We have found that these shortcomings are mainly attributed to the update equation of the particle's velocity, which is the essence of the algorithm, and there seems to be much room for improvement on this, in order to enhance the global search ability of the PSO algorithm.

In this study, inspired by the free electron model in metal conductors placed in an external electric field [25], we propose a novel variant of the PSO algorithm, called the random drift particle swarm optimization (RDPSO), and apply it to HMM training for MSA. The motivation of the RDPSO algorithm is to improve the search ability of the PSO by fundamentally modifying the update equation of the particle's velocity, instead of by merely adding some other operations into the algorithm so as to probably increase the complexity of the algorithm and its computational cost. Furthermore, in order to enhance the global search ability of the RDPSO for hard MSA problems, we incorporate a diversity control strategy into the RDPSO and propose a RDPSO with diversity-guided search (RDPSO-DGS), in which the diversity measure of the particle swarm is controlled to maintain it at a certain level, in order to prevent the premature convergence at a later stage of the search process. The performance of the RDPSO and RDPSO-DGS in learning HMMs for MSAs is tested on three benchmark data sets and compared to the performance of other learning algorithms and several well-known multiple sequence alignment programs.

In [22], we proposed the so-called diversity-maintained quantum-behaved particle swarm optimization (DMQPSO) algorithm and applied it to MSA problems. This method is based on controlling the diversity of particles' current positions by selecting different values for the constriction-expansion coefficient in the different stages of the QPSO algorithm. The diversity control method proposed in this paper is based on the diversity of the particles' personal best positions, which affects the diversity of the particles' current positions and plays a more important role in the global search of the algorithm. Another major difference between the work in this paper and in [22] is that the method in this paper is based on the newly proposed RDPSO algorithm, inspired by the free electron model, as explained above.

The remainder of the paper is organized as follows. Some preliminary knowledge on the topology and parameter learning of HMMs for MSA is described in Section 2. Section 3 presents the methods of scoring multiple sequence alignments. Section 4 describes the principles of the proposed RDPSO and RDPSO-DGS algorithms and their application to HMM learning for MSA. The experimental results

on the benchmark data sets are provided and discussed in Section 5. Some concluding remarks are offered in Section 6.

2 Hidden Markov Models for MSA

2.1 Topology of HMMs for MSA

For multiple sequence alignment, a Hidden Markov model denoted by λ consists of a set of q states $(S_1, S_2, \ldots, S_q)$ that are divided into three groups: match (M), insert (I) and delete (D) [15]. In addition, there are two special states, namely, the begin state and the end state. States are connected to each other by transition probabilities a_{ij} where $0 \le a_{ij} \le 1\ (1 \le i, j \le q)$ and $\sum_{j=1}^{q} a_{ij} = 1 (1 \le i \le q)$. A match or insert state, S_j, emits an observable symbol, v_k, from an output alphabet Σ with a probability $b_j(k)$ where $0 \le b_j(k) \le 1\ \ (1 \le j \le q,\ 1 \le k \le m)$ and $\sum_{k=1}^{m} b_j(k) = 1$ $(1 \le j \le q)$. Here m is the number of observable symbols. The delete states, the begin state and the end state do not emit observable symbols, so they are called silent states.

Starting from the begin state and until the end state, the HMM generates sequences, namely, strings of observable symbols, by making nondeterministic walks that randomly go from one state to another according to the transition probabilities. Each walk yields a path of visited states $\pi = (\pi_1, \pi_2, \ldots, \pi_p)$ and a sequence consisting of emitted observable symbols on the path. When the HMM is applied to MSA, the sequence of observable symbols is given in the form of an unaligned sequence. The goal of multiple sequence alignment is thus to find a path π which generates the best alignment. It is possible to use the forward and Viterbi algorithms to determine, $P(o \mid \lambda)$, the probability of a given sequence o generated by the HMM λ, and derive the path π with maximal probability of generating the sequence o [26].

2.2 Learning HMMs for MSA

For a given sequence o and a Hidden Markov Model λ, the goal of the learning task is to estimate the parameters, i.e., the transition and emission probabilities, of the HMM λ such that $P(o \mid \lambda)$ is maximized. The learning task is usually performed by either the Baum-Welch technique that is based on statistical re-estimation formulas [26], or by random search methods such as the simulated annealing (SA) [19] or evolutionary algorithms (EAs) [20]. Before parameter estimation, the length of the HMM should be determined. A commonly used estimate is the average length of the sequences to be aligned. After parameter estimation, a better length of the HMM can be chosen by using a heuristic method known as the model surgery [15].

The quality of the HMM needs to be evaluated in the process of parameter estimation. Generally, a log-odds (LO) score is used for this purpose, which is based on a log-likelihood score [26] given by

$$\text{Log} - \text{odds (O},\lambda) = \frac{1}{N}\sum_{i=1}^{N} \log_2 \frac{P(O_i \mid \lambda)}{P(O_i \mid \lambda_{\text{null}})}, \tag{1}$$

where $O = (O_1, O_2, \ldots, O_N)$ is the set of unaligned sequences, λ is the learned HMM, and λ_{null} is a null-hypothesis model. Here, a random model is chosen as the null-hypothesis model. The random model or the null-hypothesis model is the same for all the tested training algorithms for a given sequence set. However, different sequence sets use different null-hypothesis models since the numbers of states of the models vary with the lengths of sequences. The HMM obtained from the learning phase is considered to be a profile for the set of sequences. Thus, the unaligned sequences can be aligned by such a profile HMM λ. After the learning phase of the HMM, the final step is to interpret the learned sequences as a multiple sequence alignment.

3 Scoring the Multiple Sequence Alignments

There are two different methods for scoring the alignment obtained by the experiments. For the experiments without prior knowledge regarding the structure of the resulting alignments, the standard sum-of-pairs score (SOP) given below may be used:

$$\text{Sum-of-Pairs (SOP)} = \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} dis(l_i, l_j), \tag{2}$$

where l_i and l_j is are aligned sequences, dis is a distance metric, and N is the number of the sequences to be aligned. In this work, the widely accepted BLOSUM62 replacement matrix in [27] is used as the similarity metric, since it is widely accepted as a good general substitution matrix for MSA. To prevent the accumulation of many gaps in an alignment, an affine gap cost given in equation (3) is deducted from the sum-of-pairs function:

$$\text{Gap Cost (GC)} = \text{GOP} + n \times \text{GEP}, \tag{3}$$

where GOP is a fixed penalty for opening a gap, GEP is the penalty for extending the gap, and n is the number of gap symbols in the gap. With the gap cost calculated for each gap in each of the aligned sequences, the sum of these costs is then subtracted from the sum-of-pairs score as a penalty, and the resulting score is the one presented in the section of experimental results.

For the experiments where the reference alignment is available, a modified sum-of-pairs score (MSOP), is employed to evaluate the alignment as proposed in [28]. The reference alignment is a manually refined alignment that is believed to be of high quality. Given the test alignment of N sequences consisting of M columns, the ith column of the alignment can be denoted by $A_{i1}, A_{i2}, \ldots, A_{iN}$. For each pair of residues A_{ij} and A_{ik}, p_{ijk} is defined as follows: $p_{ijk} = 1$ if the residues A_{ij} and A_{ik} from the test alignment are aligned with each other in the reference alignment; otherwise $p_{ijk} = 0$. The column score is defined by

$$S_i = \sum_{j=1, j\neq k}^{N}\sum_{k=1}^{N} p_{ijk}. \tag{4}$$

The MSOP for the test alignment is then given by

$$\text{MSOP} = \sum_{i=1}^{M}\left(\frac{S_i}{\sum_{r=1}^{M_r} S_{ri}}\right), \tag{5}$$

where M_r is the number of columns in the reference alignment and S_{ri} is the score S_i for the rth column in the reference alignment.

It should be noted here that both of the two sum-of-pairs scores, i.e., SOP and MSOP, can be used for scoring a constructed MSA as well as for parameter learning, depending on what sort of information is available. If the prior knowledge of the resulting alignment is available, the MSOP function should be employed; otherwise, the SOP should be employed.

4 The Proposed RDPSO and RDPSO-DGS

4.1 A Brief Introduction to the PSO Algorithm

Particle swarm optimization is a population-based optimization technique inspired by the collective behavior of social organisms [29], [30], [31], [32], [33], [34], [35], [36]. In the PSO with L particles, each particle i $(1 \le i \le L)$ represents a potential solution of the given problem in an D-dimensional space, and has three vectors at the kth iteration: its current position $X_{i,k} = (X^1_{i,k}, X^2_{i,k}, \ldots, X^D_{i,k})$, its velocity $V_{i,n} = (V^1_{i,k}, V^2_{i,k}, \ldots, V^D_{i,k})$ and its personal best (pbest) position (the position giving the best objective function value or fitness value obtained so far) $P_{i,k} = (P^1_{i,k}, P^2_{i,k}, \ldots, P^D_{i,k})$. There is a vector $G_k = (G^1_k, G^2_k, \ldots, G^D_k)$, called the global best (gbest) position which is defined as the position of the best particle among all the particles in the population. For the HMM learning in MSA problems, the position of a particle is composed by the parameters of the HMM, that is, each component of the position vector is a parameter of the HMM; each component of the velocity vector represents the variation of the corresponding parameter during HMM training. The particle updates its velocity and position according to the following discrete equation:

更新速度

$$V^j_{i,k+1} = w \cdot V^j_{i,k} + c_1 r^j_{i,k}\left(P^j_{i,k} - X^j_{i,k}\right) + c_2 R^j_{i,k}\left(G^j_k - X^j_{i,k}\right), \quad (6)$$

更新位置

$$X^j_{i,k+1} = X^j_{i,k} + V^j_{i,k+1}, \quad (7)$$

for $1 \le i \le L$ and $1 \le i \le D$. Without loss of generality, if we are to solve the following minimization problem:

$$\text{Minimize } f(X), \text{ s.t., } X \in S \subseteq R^D, \quad (8)$$

where $f(X)$ is an objective function (or fitness function) continuous almost everywhere on S (the feasible space), then $P_{i,k}$ and G_k can be updated by equations

$$P_{i,k} = \begin{cases} X_{i,k} & \text{if } f(X_{i,k}) < f(P_{i,k-1}) \\ P_{i,k-1} & \text{if } f(X_{i,k}) \ge f(P_{i,k-1}), \end{cases} \quad (9)$$

$$G_k = P_{g,k}, \quad (10)$$

where

$$g = \arg \min_{1 \le i \le L} [f(P_{i,k})]. \quad (11)$$

In equation (6), c_1 and c_2 are called acceleration coefficients and the parameter w is known as the inertia weight, which can be adjusted to balance the exploration and the exploitation of the algorithm [30]. The parameters $r^j_{i,k}$ and $R^j_{i,k}$ are the sequences of two different random numbers distributed uniformly on (0, 1), which are denoted by $r^j_{i,k}, R^j_{i,k} \sim U(0,1)$. Generally, the value of $V^j_{i,k}$ is restricted within the interval $[-V_{\max}, V_{\max}]$.

4.2 Random Drift Particle Swarm Optimization

It was demonstrated by a trajectory analysis [36] that the convergence of the PSO algorithm may be achieved if each particle converges to its local focus, $p_{i,k} = (p^1_{i,k}, p^2_{i,k}, \ldots, p^D_{i,k})$ defined at the coordinates

自己最好 大家最好

$$p^j_{i,k} = \phi^j_{i,k} P^j_{i,k} + \left(1 - \phi^j_{i,k}\right) G^j_k, \quad (12)$$

where $\phi^j_{i,k} = c_1 r^j_{i,k} / (c_1 r^j_{i,k} + c_2 R^j_{i,k})$ with regard to the random numbers $r^j_{i,k}$ and $R^j_{i,k}$ in equations (6) and (7). Since the acceleration coefficients c_1 and c_2 are generally set to be equal, i.e., $c_1 = c_2, \phi^j_{i,k}$ is a sequence of uniformly distributed random numbers on (0,1), varying with k for each i and j. Consequently, equation (12) can be restated as

p落在於P和G所圍出的矩形中

$$p^j_{i,k} = \phi^j_{i,k} P^j_{i,k} + \left(1 - \phi^j_{i,k}\right) G^j_k, \phi^j_{i,k} \sim U(0,1), \quad (13)$$

which implies that $p_{i,k}$ is located randomly within the hyperrectangle with $P_{i,k}$ and G_k being the two ends of its diagonal, and it moves following the $p_{i,k}$ and G_k. In practice, as the particles are converging to their own local focuses, their current positions, personal best positions, local focuses and the global best position are all converging to one point, which leads the PSO algorithm to converge. It can be found that the particle's directional movement toward $p_{i,k}$ is similar to the drift motion of an electron in metal conductors in an external electric field.

However, according to the free electron model [25], besides the directional motion caused by the electric field, the electron is also in a thermal motion, which appears to be a random movement. Therefore, the movement of the electron's movement is the superimposition of a random motion and a drift motion, and its velocity $V = VR + VD$, where VR and VD represent the velocities of the random motion and the drift motion, respectively. The random motion is essentially a thermal motion, which exists even without the external electric field as long as the temperature is greater than absolute zero, whereas the drift motion is a directional movement in the opposite direction of the external electric field. The overall effect of the electron's movement is that the electron careens towards the location of the minimum potential energy. Therefore, it is obvious that the movement of the electron is very similar to the process of finding the minimum solution of a minimization problem, with the position of the electron being analogous to a candidate solution and the potential energy function to the objective function of the problem.

Inspired by the above facts, we assume that the particle in PSO behaves like an electron moving in a metal conductor in an external electric field, and the movement of the particle is the superposition of the thermal motion and the drift motion towards $p_{i,k}$. Accordingly, the jth component of the particle's velocity can be expressed by $V^j_{i,k} = VR^j_{i,k} + VD^j_{i,k}$, where $VR^j_{i,k}$ and $VD^j_{i,k}$ denotes the jth components of the velocities of the random motion and drift motion, respectively.

At the $(k+1)$th iteration, the velocity of random motion $VR^j_{i,k+1}$ is assumed to follow the Maxwell velocity distribution law. Consequently, $VR^j_{i,k+1}$ essentially follows a normal distribution (i.e., Gaussian distribution) whose probability density function is given by

$$f_{VR^j_{i,k+1}}(v) = \frac{1}{\sqrt{2\pi}\sigma^j_{i,k+1}} e^{\frac{-v^2}{2(\sigma^j_{i,k+1})^2}}, \quad (14)$$

where $\sigma^j_{i,k+1}$ is the standard deviation of the distribution. Thus, a stochastic simulation can lead to the following expression for $VR^j_{i,k+1}$:

$$VR^j_{i,k+1} = \sigma^j_{i,k+1}\phi^j_{i,k+1}, \quad (15)$$

where $\varphi^j_{i,k+1}$ is a random number with a standard normal distribution, i.e., $\phi^j_{i,k+1} \sim N(0,1)$. $\sigma^j_{i,k+1}$ must be determined in order to calculate $VR^j_{i,k+1}$. An adaptive strategy is adopted for $\sigma^j_{i,k+1}$:

$$\sigma^j_{i,k+1} = 2\alpha\,|\,C^j_k - X^j_{i,k}\,|, \quad (16)$$

where $C_k = (C^1_k, C^2_k, \ldots, C^D_k)$ is known as the mean best (mbest) position, defined by the mean of the personal best positions of all the particles, namely:

$$C^j_k = (1/L)\sum_{i=1}^{L} P^j_{i,k}, \quad (1 \le j \le D). \quad (17)$$

Thus, equation (15) can be restated as

$$VR^j_{i,k+1} = \alpha \mid C^j_k - X^j_{i,k} \mid \phi^j_{i,k+1}, \quad (18)$$

where $\alpha > 0$ is an algorithmic parameter called the thermal coefficient.

The velocity of the drift motion $VD^j_{i,k+1}$ can be expressed in many forms. However, in this work, a simple linear expression is adopted for $VD^j_{i,k+1}$:

$$VD^j_{i,k+1} = \beta\left(p^j_{i,k} - X^j_{i,k}\right), \quad (19)$$

where $\beta > 0$ is another algorithmic parameter. If there is only drift motion, that is, $V^j_{i,k+1} = VD^j_{i,k+1}$, the above discrete equation can ensure the particle moves towards $p^j_{i,k}$ in the jth dimension at the $(k+1)$th iteration.

With the above specifications, a novel set of update equations for the particle can be obtained as

$$V^j_{i,k+1} = \alpha \mid C^j_k - X^j_{i,k} \mid \phi^j_{i,k+1} + \beta\left(p^j_{i,k} - X^j_{i,k}\right), \quad (20)$$

thermal drift

$$X^j_{i,k+1} = X^j_{i,k} + V^j_{i,k+1}, \quad (21)$$

where α and β are called thermal coefficient and drift coefficient, respectively, and $p^j_{i,k}$ is given by equation (13). The PSO with equations (20) and (21) is named the random drift particle swarm optimization, the procedure of which is outlined in Algorithm 1.

In the RDPSO algorithm, besides the population size, α and β are two important user-specified parameters, which can be adjusted to balance the global and local search of the particles. Moreover, the value of $V^j_{i,k}$ is generally restricted in the interval $[-V_{\max}, V_{\max}]$ when the algorithm is running.

Algorithm 1: The RDPSO-DGS algorithm

Step 0: Initialize the current positions of all the particles randomly, set the *pbest* position of each particle to be its current position, and find the *gbest* position;
Step 1: Set k=0;
Step 2: While the termination condition is not met, **do** the following steps;
Step 3: Set k=k+1 and compute the mean best position C_k;
Step 4: Measure d_k according to equation (22);
Step 5: From $i = 1$, execute the following steps;
Step 6: Evaluate the objective function value $f(X_{i,k})$, and update $P_{i,k}$ and G_k according to equations (9) and (10), respectively;
Step 7: If $d_k < d_{low}$, execute Steps 8 and 9; or else go to Step 10;
Step 8: For each j, execute the mutation operation $P'^j_{g,k} = P^j_{g,k} + \delta A\varepsilon$;
Step 9: Set $P_{g,k} = P'_{g,k}$ and $G_k = P'_{g,k}$, and update the fitness value of $P_{g,k}$ by $f(P'_{g,k})$;
Step 10: Update the jth $(1 \le j \le N)$ components of velocity and current position of particle i according to equations (20) and (21) with $p^j_{i,k}$ given by equation (13);
Step 11: Set i=i+1, and return to Step 5 **until** $i = L$;
Step 12: Return to Step 2;

4.3 RDPSO with Diversity-Guided Search

The major issue with PSO and other evolutionary algorithms in multi-modal optimization is their premature convergence, which results in significant performance loss and sub-optimal solutions. The main reason for this is that the collectiveness of particles leads to fast information flow between particles and, in turn, rapid decline of the diversity of the particle swarm. At the beginning of the search, after initialization, the diversity of the population is high. With the progress of evolution, the convergence of the particle makes the diversity decline, which, accordingly, enhances the local search ability (exploitation) but weakens the global search ability (exploration) of the algorithm. At early or middle stage of the evolution, the decline of the diversity is desirable for the particle swarm to search effectively. However, after middle or at later stage, the particles may converge into such a small region that the diversity of the swarm is very low and further search is difficult. At that moment, if the particle with the global best position is at a local optima or sub-optima, stagnation or premature convergence occurs. Although the RDPSO algorithm may have stronger global search ability than the canonical PSO, the diversity loss of the particle swarm is also inevitable due to the collectiveness of the particle swarm.

One way to circumvent the above shortcoming is to employ a diversity control strategy to prevent the premature convergence, as in [37] and [38]. Here, a diversity-guided search is incorporated into the RDPSO algorithm, in order to enhance the global search ability of the algorithm even further.

In the RDPSO with diversity-guided search, the diversity measure at the kth iteration, denoted as d_k, is given by the average euclidean distance from the particle's current

position to the centroid of the swarm, as in [35]. That is,

$$d_k = \frac{1}{L \cdot A} \sum_{i=1}^{L} \sqrt{\sum_{j=1}^{D} \left(X_{i,k}^{j} - \overline{X}_{k}^{j}\right)^2}, \tag{22}$$

where $\overline{X}_k^j$ is given by $\overline{X}_k^j = (1/L)\sum_{i=1}^{L} X_{i,k}^j$. In (22), A denotes the length of the longest diagonal in the search space, and D is the dimensionality of the problem. A lower bound d_{low} is set for d_k to prevent the diversity from constantly decreasing on the course of the search. If the diversity measure d_k drops below d_{low}, it will be controlled in such a way to increase it until it is larger than d_{low}. A mutation operation exerted on the global best particle $P_{g,k}$ plays such a role of diversity increasing. Once d_k is smaller than d_{low}, $P_{g,k}$ is mutated as

$$P'^{j}_{g,k} = P^{j}_{g,k} + \delta A \varepsilon, \tag{23}$$

for $1 \le j \le D$. $P'_{g,k} = (P'^{1}_{g,k}, \ldots, P'^{j}_{g,k}, \ldots, P'^{D}_{g,k})$ is the new position of the global best particle after mutation, which replaces the original $P_{g,k}$. Then the *gbest* position is also set to $G_k = P'_{g,k}$, but its fitness value remains unchanged. In equation (23), ε is a random number with standard normal distribution varying with k and j, δ is called the mutation scaling coefficient, which is a user-specified algorithmic parameter. It should be noted that, when the mutation operation is implemented, the displacement of the global best particle will generally increase the value of $|P_{g,k}^{j} - P_{i,k}^{j}|$ and pull the mean best position C_k away from its original location, consequently making the particles more volatile and increasing the diversity measure d_k. Besides, the mutation helps the global best position G_k jump out of sub-optimal or local optimal solutions, which in turn could guide the whole particle swarm to search more promising areas. Algorithm 1 outlines the procedure of the RDPSO-DGS algorithm.

It can be seen that the RDPSO-DGS runs in convergence mode, with the diversity measure declining during most of the iterations. Only when the diversity declines below the lower bound d_{low}, do the particles explode, due to the mutation operation. This explosion process is transitory, and once the diversity is over the threshold d_{low}, the whole particle swarm returns to convergence mode again. As for d_{low}, our preliminary experimental results for several widely used benchmark functions showed that setting $d_{low} \approx 10^{-4}$ may lead to good algorithmic performance in general. The scaling coefficient of the mutation in RDPSO is another important parameter whose value selection will be discussed in the next section.

4.4 Learning HMMs for MSA with RDPSO or RDPSO-DGS

For a given sequence set, when the RDPSO or RDPSO-DGS is used to perform HMM learning for multiple sequence alignment, the length of the HMM is kept constant during the learning process and only the parameters of the HMM are optimized. As indicated in Section 2, the length of the HMM is set to the average length of the sequences in the given sequence set. After the learning process, a better length of the HMM can be chosen by the *model surgery*. That means the length of HMM varies with the different sequence sets, but it would be fixed for a given sequence set. The parameters to be estimated include the transition and emission probabilities. Thus, a candidate solution represented by the position of a particle is a real-valued vector containing l transitions and m emission probabilities, which means that the dimension of the search space for the HMM training is $D = l + m$.

On the course of the learning process, a copy of the current positions of all the particles is created at each iteration of the RDPSO or RDPSO-DGS. The position of each particle in this copy is normalized so that the constraints on the transition and emission probabilities described in Section 2 can be satisfied. We denote the normalized copy of the particle swarm at the kth iteration by $X'_k = [X'_{1,k}, X'_{2,k}, \ldots, X'_{L,k}]$. $X'_{i,k}$ $(1 \le i \le L)$ in X'_k is the normalized position of particle i and is evaluated either by the log-odds in equation (1) or by the sum-of-pairs score in equation (2) or (5), which are employed as the fitness value function for the evaluation of the HMM particle. The procedure of HMM learning with RDPSO or RDPSO-DGM is described in Algorithm 2.

Algorithm 2: Learning HMMs for MSA with RDPSO or RDPSO-DGS

Step 0: Initialize the current positions of all the particles randomly, set the *pbest* position of each particle to be its normalized current position, and find the *gbest* position;
Step 1: Set k=0;
Step 2: While the termination condition is not met, **do** the following steps;
Step 3: Set k=k+1 and compute the mean best position C_k;
Step 4: Measure d_k according to equations (22) (for HMM learning with RDPSO-DGS);
Step 5: Copy the current positions of all the particles to X'_k;
Step 6: Normalize X'_k;
Step 7: From $i = 1$, execute the following steps;
Step 8: Evaluate the objective function value $f(X'_{i,k})$, and if $f(X'_{i,k}) > f(P_{i,k})$, then $P_{i,k} = X'_{i,k}$ and update G_k accordingly;
Step 9: Execute Step 10 in Algorithm 1 when learning HMM with the RDPSO, or execute Steps 7 to 10 in Algorithm 1 when learning HMM with the RDPSO-DGS, in order to update the current position $X_{i,k}$.
Step 10: Set i=i+1, and return to Step 7 **until** $i = L$;
Step 11: Return to Step 2;

4.5 MSA with the Trained HMM

After the learning process, the output *gbest* position of the particle swarm represents a set of optimized parameters of the HMM, including the transition and emission probabilities. The learned HMM can be considered as a profile for the set of given sequences, and can be used to align multiple sequences, in conjunction with the Viterbi algorithm [25]. The resulting alignment of multiple sequences is then evaluated according to the sum-of-pairs scores described in Section 3.

5 Experimental Results and Discussions

5.1 Benchmark Data Sets

In order to evaluate the performance of the proposed algorithms, the RDPSO, RDPSO-DGS, PSO and BW algorithm were used to learn HMMs for MSA problems. Two benchmark data sets, including (a) a simulated nucleotide data set, (b) an amino-acid data set from the benchmark

TABLE 1
The 12 Benchmark Data Sets from BAliBASE

	Amino-acid Sequences	Number of sequences	Min. & Max. LESQ (min, max)	Identity (%)
Short sequence sets	1aboA	5	(49,80)	<25
	1idy	5	(49,58)	<25
	451c	5	(70,87)	20-40
	1krn	5	(66,82)	>35
Medium sequence sets	1bbt3	5	(149,192)	<25
	kinase	5	(263,276)	<25
	1pii	4	(247,259)	20-40
	5ptp	5	(222,245)	>35
Long sequence sets	gal4	5	(335,395)	<25
	1ajsA	4	(258,387)	<25
	glg	5	(438,486)	20-40
	1tag	5	(806,928)	>35

alignment database (BaliBASE) were used in the test. The performance of the above four methods were also compared with the ClustalW program and the more recently developed MAFFT (L-iNS-I method) program.

5.1.1 The Simulated Nucleotide Data Set

Nucleotide sequences (20 sequences with each having approximately 300 characters) generated by using the program Rose [39] (using the model proposed by Juke and Cantor [40] and taking the mean substitution rate = 0.013 and the insert/delete probability = 0.03) are obtained as follows. A randomly generated root sequence (of length 500) was evolved on a random tree to yield sequences of 'low' or 'high' mean divergences, i.e., with an average number of substitutions per site of 0.5 or 1.0, respectively. Furthermore, the insertion/deletion length distribution was set to 'short' (frequencies of gaps of length 1-3 = 0.8, 0.1, 0.1) or 'long' (frequencies of gaps of length 1-7 = 0.3, 0.2, 0.1, 0.1, 0.1, 0.1, 0.1). Hence the four 'mean divergence-mean gap length' combination of conditions tested here are 'low-short' 'low-long', 'high-short', and 'high-long'. Fifty random sequences were generated under each of the four combinations of settings, and the average performance was recorded. For this data set, the sum-of-pairs given by equation (2) was employed to score the resulting alignments.

5.1.2 The Amino-Acid Data Sets

The second benchmark data set was extracted from the BAliBASE (benchmark alignment database) database at http://www.cs.nmsu.edu/~jinghe/CS516BIOINFO/Fall05/Bali BASE/align_index.htmlUTH. It contains several manually refined multiple sequence alignments, specifically designed for the evaluation and comparison of multiple sequence alignment methods. Twelve sequence sets were selected from the first reference set from the BALIBASE database. Table 1 lists the names of selected sequence sets, the number of sequences in each set, the minimum and maximum lengths of the sequences (denoted as LSEQ) in each set, and the identity of the sequences, represented by the percent of identical residues between the sequences. These sequence sets were selected in order to cover different length scopes (according to the minimum and maximum lengths of the sequences and different identities. Four sequence sets were selected from the short sequence sets in the database, four were from the medium ones in the database, and four were from the long ones in the database. Each group of sequence sets consist of sequence sets with low, medium and high identities respectively. In short sequence sets, laboA and lidy have low identities, 451c has medium identity and lkrn has high one. In medium sequence sets, 1bbt3 and kinase has low identities, 1pii has medium identity and 5ptp has high identity. In the long sequence sets, gal4 and ljasA have low identities, glg has medium identity and ltag has high one. These sequence sets with a variety of lengths and identities can provide a comprehensive evaluation to the proposed methods. Since the reference alignments for these data sets are available, the modified sum-of-pairs (MSOP) score in equation (5) was used to train the HMM and evaluate the quality of the resulting test alignments.

5.2 Experimental Settings

5.2.1 Algorithmic Parameter Setting

For PSO, RDPSO, and RDPSO-DGS, the population size can be set from 20 to 80, as most of the literature on PSO recommend. In the experiments of this study, each of the RDPSO, RDPSO-DGS and PSO used 20 particles (i.e., population size was 20) which were initialized randomly with a uniform distribution within the search scope $[0,1]^D$. The upper limit of the velocity $V_{\max}$ for each PSO variants, as recommended by most of the literature, is generally set to be $a \cdot (X_{\max} - X_{\min})$, where $0 < a \leq 1$ is a scaling coefficient and $(X_{\max} - X_{\min})$ is the width of search scope in each dimension. In our experiments for multiple sequence alignment, a was set to 1 and, thus, $V_{\max} = 1$ since $X_{\max} - X_{\min} = 1 - 0 = 1$.

For the PSO algorithm, we employed a version of the standard PSO (SPSO), defined as a combination of the PSO with constriction factor and the ring neighborhood topology [32]. The SPSO has been shown to outperform many recently proposed PSO variants [33]. In our experiments, the parameters of the standard PSO were determined according to published recommendations [32], that is, the constriction factor χ was set to 0.729, and both acceleration coefficients were set as $c_1 = c_2 = 2.05$.

For the RDPSO and RDPSO-DGS, the thermal and drift coefficients are two important algorithmic parameters corresponding to the inertia weight and acceleration coefficients in PSO. These two parameters can be tuned to balance the local and global search of the particle. Larger thermal and drift coefficients mean stronger global search ability of the particle, while smaller ones imply stronger local search ability. A theoretical analysis of the two coefficients' influence on the convergence behavior of the particle and the algorithm is out of the scope of this paper and will be our future task. However, our preliminary experiments on several benchmark functions showed that decreasing the thermal coefficient linearly from a large value to a small one, and setting the drift coefficient at a fixed value during the search process, can lead to a stable performance of the algorithm. Furthermore, it was also found, through preliminary experiments, that it can result in generally good algorithmic performance if we decrease the thermal coefficient α linearly from 1.0 to 0.5 and fix the drift coefficient β at 1, during the

TABLE 2
HMM Log-Odds for the Nucleotide Data Set and the Execution Time of Each Algorithm

Algorithms	Simulated Nucleotide	Average LO Score (St. Dev.)	Normalized Average LO Score	Execution Time (h)	Simulated Nucleotide	Average LO Score (St. Dev.)	Normalized Average LO Score	Execution Time (h)
BW	Low-short	394.6	-1.1785	**0.0135**	High-short	186.3	-0.8407	**0.0115**
SPSO		419.5 (7.42)	-0.4022	1.4501		195.6(2.31)	-0.6962	1.7726
RDPSO		447.6 (9.63)	0.4739	1.4478		253.8(3.45)	0.2082	1.7966
RDPSO-DGS		**467.9(3.95)**	**1.1068**	1.4518		**325.9(3.84)**	**1.3286**	1.8176
BW	Low-long	226.5	-0.7191	**0.0126**	High-long	280.8	-0.5968	**0.0124**
SPSO		215.8 (8.26)	-0.8430	1.4304		256.4(8.52)	-1.0904	1.7808
RDPSO		310.7 (4.68)	0.2555	1.4316		345.8(9.73)	0.7182	1.7865
RDPSO-DGS		**401.5 (2.86)**	**1.3065**	1.4404		**358.2(7.21)**	**0.9690**	1.7915

search process. These parameter configurations were used for the RDPSO and RDPSO-DGS in our experiments for multiple sequence alignment.

For the RDPSO-DGS, d_{low} is an important parameter that determines the level at which the diversity is maintained in order to balance the exploration and the exploitation of the swarm at the later stage of search. If the value of d_{low} is set to be too large, the RDPSO-DGS can have stronger global search ability, but its local search ability is weakened. On the contrary, if it is too small, the algorithm may have better local convergence but its global search ability will not be improved significantly compared to the RDPSO algorithm. Our preliminary experiments suggested that a value of d_{low} within the interval $[10^{-3}, 10^{-6}]$ could lead to satisfactory algorithmic performance of the algorithm. In the current study, d_{low} was set to be 10^{-4} which appeared to generate good performance of the RDPSO-DGS on average, although better results may be obtained by using other values of d_{low} as shown in the experiments. Another important parameter in the RDPSO-DGS is the mutation scaling coefficient δ. Since the multiplication of δ and A gives the standard deviation of the mutation, a larger value of δ results in a greater displacement of the global best particle and, thus, a larger diversity gain after mutation, while a smaller δ means a smaller increase of the diversity measure. In order to balance the global and local search of the algorithm, the value of δ should be selected properly. From our preliminary experiments, we recommend that δ should take value within $[10^{-2}, 10^{-8}]$, and varying with the dimensionality of the problems. In the experiments of the current study, δ was set to be 10^{-5}. In addition, since the search scope in each dimension for each instance is $[0, 1]$, A is numerically equal to $\sqrt{D}$ according to its definition, where is D the dimension of the search space, i.e., the number of HMM parameters. For ease of reference, we list the algorithmic parameter settings in detail for the SPSO, RDPSO and RDPSO-DSG, as follows:

SPSO: population size $L = 20$; constriction factor $\chi = 0.729$; $c_1 = c_2 = 2.05$; ring neighborhood topology was used; $V_{\max} = 1.0$.

RDPSO: population size $L = 20$; α decreased linearly from 1.0 to 0.5; $\beta = 1$; $V_{\max} = 1.0$.

RDPSO-DGS: population size $L = 20$; α decreased linearly from 1.0 to 0.5; $\beta = 1$; $d_{low} = 0.0001$; $\delta = 10^{-5}$; $V_{\max} = 1.0$.

5.2.2 Other Experimental Configurations

For nucleic acid sequences, the 'swap' substitution score table from ClustalW 1.81 version was used as the distance metric in the sum-of-pairs score function, and a GOP of 15 and a GEP of 7 were used as penalties. For the amino-acid data set, the BLOSUM62 replacement matrix in [27] was used as the distance metric in the sum-of-pairs score function and GOP and GEP were set to be 11 and 2, respectively. These parameter values were configured as in [21].

For each set of sequences in the three data sets, the experiments for HMM learning with the four algorithms were performed using log-odds score and sum-of-pairs score as the objective function. For each learning experiment, the BW is a deterministic algorithm, to which a fixed initial HMM was input, and from which a fixed final HMM was output. The other three learning algorithms, namely the SPSO, RDPSO and RDPSO-DGS, are random search methods, each of which performed 20 runs for each learning task with each run executing 1,000 iterations.

5.3 Experimental Results

After the RDPSO-DGS, RDPSO, SPSO and BW algorithms performed the HMM learning tasks on each data set, the average final fitness values (i.e., log-odds scores or sum-of-pairs scores) and their standard deviations over 20 runs of each algorithm were recorded and compared.

To investigate the convergence property of the algorithm, the fitness value at each iteration of the learning process averaged over 20 runs of each algorithm were also recorded and visualized. The average execution time over 20 runs of parameter learning for each algorithm was also recorded for the purpose of comparing the computational costs among the tested algorithms.

In order to evaluate the algorithmic performance on a uniform scale, we normalized the average scores of each algorithm on each sequence set. The normalized score is defined by

$$\text{Normalized Score (NS)} = (S_i - \overline{S})/\sigma_S, \tag{24}$$

where S_i is the score, $\overline{S}$ is the mean of the scores and σ_S is the standard deviation of the scores. The normalized scores can be summed or averaged across different data sets for an overall performance comparison.

5.3.1 Results for the Simulated Nucleotide Data Set

The experimental results, including average scores, standard deviation, normalized scores, and average execution time, for the HMM learning on the simulated nucleotide data set are provided in Tables 2 and 3. Table 2 lists the statistical results with the log-odds score used as the quality

TABLE 3
SOP Scores for the Nucleotide Data Set and the Execution Time of Each Algorithm

Algorithms	Simulated Nucleotide	Average SOP (St. Dev.)	Normalized Average SOP Score	Execution Time (h)	Simulated Nucleotide	Average SOP (St. Dev.)	Normalized Average SOP Score	Execution Time (h)
ClustalW	Low-short	2514	0.0299	--	High-short	4316	-0.4405	--
MAFFT		2743	0.6830	--		**4459**	**1.6371**	--
BW		2024.7	-1.3654	0.0143		4270.2	-1.1058	0.0159
SPSO		2157.8 (23.17)	-0.9858	1.9671		4294.7(13.57)	-0.7499	1.9234
RDPSO		2643.1 (20.05)	0.3981	1.9741		4351.9(13.38)	0.0811	1.9346
RDPSO-DGS		**2938.4(8.49)**	**1.2402**	2.0340		4386.1(12.47)	0.5780	1.9531
ClustalW	Low-long	8760	-0.3730	--	High-long	7781	0.0303	--
MAFFT		8463	-0.7870	--		7563	-0.5468	--
BW		8623.9	-0.5628	0.0140		7418.2	-0.9301	0.0156
SPSO		8438.7 (13. 36)	-0.8209	1.9457		7538.5(22.86)	-0.6116	2.0159
RDPSO		9876.8 (16.22)	1.1836	1.9650		7848.8(29.69)	0.2097	2.1268
RDPSO-DGS		**10003.4(14.37)**	**1.3601**	1.9762		**8467.9(23.96)**	1.8485	2.2020

measure, namely the fitness value, for HMM learning. The results for each method (and for each sequence set) include the average log-odds (LO) score of the resulting alignments obtained by the final HMMs over 20 runs of training, the standard deviation (in brackets) of the LO scores over 20 runs of training, the normalized average LO score and the average execution over 20 runs of training. Here, each normalized average LO score was obtained by equation (24) for a certain sequence set, where S_i is the average LO score of each method for that sequence set, $\overline{S}$ is the mean of the average LO scores of all the methods for that sequence set, and σ_S is the standard deviation of the average LO scores of all the methods for the sequence set. The BW is a deterministic algorithm so that it was tested once and has no standard deviation for the LO scores. Among all the learning algorithms, the RDPSO-DGS was shown to obtain the learned HMMs that had the best average LO score (or normalized average LO score) over 20 runs on each nucleotide sequence set. It is also shown that the RDPSO had the second best performance in HMM learning for each sequence set. The SPSO obtained the average and normalized average scores better than the BW algorithm, which had the worst performance in HMM training for each nucleotide sequence set among all tested learning algorithms. It is shown in Table 3 that the BW took much less time than the SPSO, RDPSO and RDPSO-DGS to train HMMs. However, the BW is a local optimization technique and is not efficient in HMM training, particularly when the number of states of the HMM is large and the training problem is very difficult to solve. Therefore, global search techniques, such as the RDPSO and RDPSO-DGS, are promising tools for that purpose although they need more execution time.

Table 3 lists the average and the standard deviation (in brackets) of the sum-of-pairs scores of the resulting alignments obtained by the final HMMs over 20 runs of training of each algorithm, the normalized average SOP score, and the execution time when the sum-of-pairs score was employed as the fitness value for HMM learning. The table also gives the SOP scores of alignments produced by the ClustalW and MAFFT (L-iNS-I method) programs, for comparison reasons. Since the ClustalW and MAFFT programs are progressive alignment methods that do not involve HMM learning and can achieve multiple sequence alignment within 1 minute, we only recorded and compared the computational consumptions of the tested optimization algorithms in HMM learning for MSAs. The normalized average SOP score of each method was obtained by equation (24) for a certain sequence set, where S_i is the average SOP score of each method for that sequence set, $\overline{S}$ is the mean of the average SOP scores of all the methods for that sequence set, and σ_S is the standard deviation of the average SOP scores of all the methods for the sequence set. It is clear that the RDPSO-DGS obtained better average SOP scores and normalized average SOP scores than other methods, including ClustalW and MAFFT, for Low-short, Low-long, High-long sequence sets. For the Low-short sequence set, the MAFFT program obtained the alignment with the second best SOP score (or normalized SOP score). The SOP score of the alignment obtained by MAFFT is the second best for this sequence set, and the SOP score of the alignment generated by the HMM trained with the RDPSO is the third best. For Low-long sequence set, the alignment with the second best SOP score was obtained by the HMM learned by the RDPSO. The alignment yielded by ClustalW has the next best SOP score and normalized SOP score. For this sequence set, the MAFFT program is showed to align with the worst SOP score. For the High-short sequence set, MAFFT produced the alignment with the best SOP score, and the HMM trained by the RDPSO-DGS generated the final alignment with the second best SOP score. For the High-long sequence set, the HMMs trained by the RDPSO-DGS and RDPSO obtained the best and the second best alignment in terms of SOP scores. ClustalW yielded the alignment with the third best SOP score (or normalized SOP score). Among the training algorithms for HMMs, the BW took less computational time for HMM training than the other algorithms. However, it is worth using the RDPSO-DGS, RDPSO and SPSO, even if it takes more time, as the obtained HMM can align the sequences more accurately.

Fig. 1 illustrates the convergence process over 20 runs of each algorithm in learning the HMMs for Low-Short nucleotide sequence set, with the log-odds scores and SOP scores being the objective function values, respectively. As shown by the figure, the BW algorithm converged at the fastest speed to the local optimal solution. The convergence of the SPSO was more rapid than that of the RDPSO and RDPSO-DGS, but it was prone to encounter premature convergence. In contrast, the RDPSO and its improved version had slower convergence speed at the early stage of the learning process, which implies that the algorithms searched globally. At the

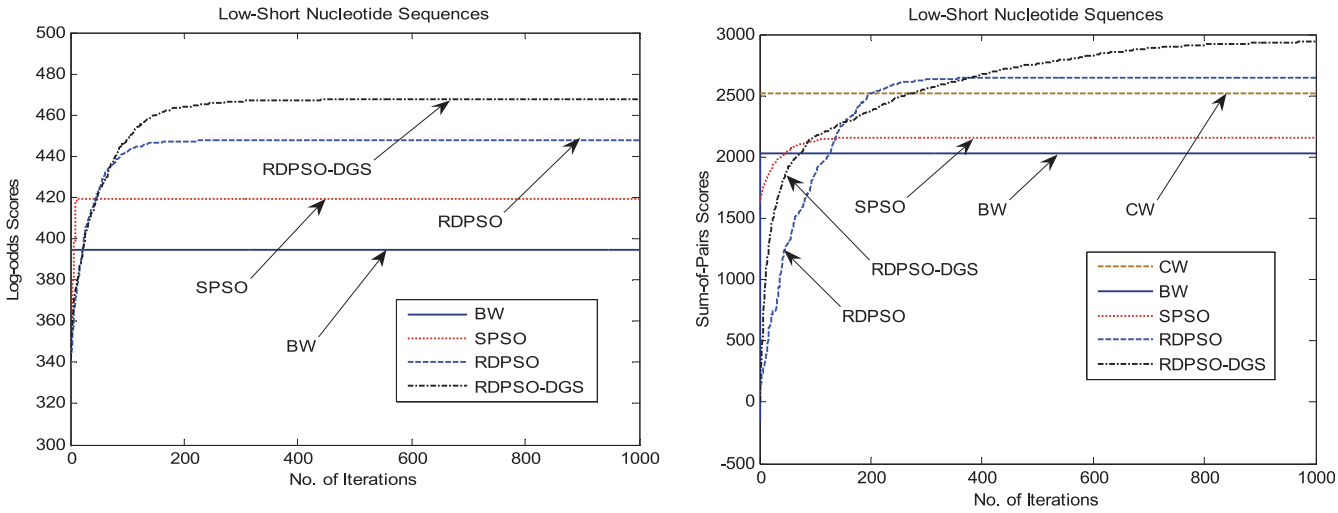

Fig. 1. Average fitness values found by the algorithms during the HMM learning process for low-short nucleotide sequences, with the fitness values being the log-odds scores and the sum-of-pairs scores.

later stage of the search, the RDPSO-DGS maintained its diversity at a certain level to prevent the particles from stagnation so that it can find better solutions than its competitors.

5.3.2 Results for Amino-Acid Data Sets

The experimental results for the amino-acid data sets from the BaliBASE database are presented in Tables 4 and 5. Table 4 lists the results when the log-odds score was used as the objective function. The results include the average and the standard deviation of the LO scores of the resulting alignments obtained by the final HMMs over 20 runs of each training algorithm for each sequence set, the normalized average LO score of each method for a sequence set, and the average execution time for each training run. Here, the normalized average LO score of each method for each sequence set was calculated in a similar way as in Tables 2 and 3. It can be observed in the table that the HMMs trained by the RDPSO-DGS obtained the alignments with the best average LO scores and, consequently, best normalized average LO scores for all of the 12 sequence sets. The RDPSO algorithm, as can be seen from the results, had the second best performance in the HMM learning for all the sequence sets. The BW algorithm, just as in the experiments for the nucleotide data sets, performed the worst in the HMM training for most of the sequence sets, but consumed the least computational time. Although the RDPSO-DGS, RDPSO and SPSO consumed more computational time, they are far more efficient in HMM training for MSA than the BW.

Table 5 provides the average of the modified sum-of-pairs (MSOP) scores of the resulting alignments obtained by the HMMs from the 20 runs of training of each algorithm

TABLE 4
HMM Log-Odds Scores for the BAliBASE Test Sets and the Execution Time of Each Algorithm

Algorithms	Sequence Sets	Average LO Score (St. Dev.)	Normalized Average LO Score	Execution Time (h)	Sequence Sets	Average LO Score (St. Dev.)	Normalized Average LO Score	Execution Time (h)
BW	1aboA	46.3814	-1.2386	**3.9444e-004**	1pii	213.0459	-1.2915	**0.0020**
SPSO		64.3318 (0.5268)	-0.3651	0.0095		274.7114(0.8156)	-0.2637	0.2045
RDPSO		85.1728 (0.7395)	0.6491	0.0102		327.4134(0.7751)	0.6146	0.2138
RDPSO-DGS		**91.4526 (0.4816)**	0.9546	0.0108		**346.9764(0.3218)**	0.9406	0.2215
BW	1idy	42.0576	-1.2377	**3.8056e-004**	5ptp	266.5928	-1.0125	**0.0019**
SPSO		60.5831 (0.3642)	-0.2611	0.0126		315.2782(0.9246)	-0.5872	0.1958
RDPSO		72.6936 (0.8358)	0.3774	0.0132		426.1276(0.6825)	0.3812	0.2084
RDPSO-DGS		**86.8073 (0.3641)**	1.1214	0.0137		**521.9736(0.3261)**	1.2185	0.2216
BW	451c	68.3522	-1.2226	**5.3056e-004**	gal4	347.2819	-0.9662	**0.0040**
SPSO		88.0682 (0.5219)	-0.3258	0.0115		392.4457(0.7290)	-0.5803	0.3568
RDPSO		105.2954(0.6426)	0.4578	0.0119		490.3946(0.5292)	0.2566	0.3589
RDPSO-DGS		**119.2075(0.7452)**	1.0906	0.0125		**611.3362(0.4395)**	1.2899	0.3603
BW	1krn	69.0222	-1.1389	**4.1667e-004**	1ajsA	326.4896	-0.9719	**0.0030**
SPSO		80.4873(0.8257)	-0.5254	0.0109		376.5527(0.7614)	-0.5747	0.2518
RDPSO		103.1493(0.8119)	0.6873	0.0118		481.6447(0.5147)	0.2589	0.2565
RDPSO-DGS		**108.5623(0.5836)**	0.9770	0.0127		**611.3362(0.4395)**	1.2877	0.2588
BW	1bbt3	172.3816	-0.7348	**0.0012**	glg	380.7306	-0.8250	**0.0060**
SPSO		170.3738(0.8148)	-0.7828	0.1509		397.2433(0.8529)	-0.6750	1.9125
RDPSO		210.1405(0.6831)	0.1690	0.1581		487.1988(0.5394)	0.1422	1.9178
RDPSO-DGS		**259.4206(0.4806)**	1.3486	0.1617		**621.0127(0.4523)**	1.3578	1.9224
BW	kinase	214.9693	-0.8158	**0.0025**	1tag	729.3726	-0.9554	**0.0138**
SPSO		214.6612(0.7931)	-0.8184	0.2320		765.9068(0.9504)	-0.6363	2.7740
RDPSO		360.2450(0.6798)	0.4113	0.2404		877.6902(0.7478)	0.3401	2.7980
RDPSO-DGS		**456.3285(0.3672)**	1.2229	0.2516		**982.0326(0.4528)**	1.2516	2.8368

TABLE 5
MSOP Scores for the BAliBASE Test Sets and the Execution Time of Each Experiment

Algorithms	Sequence Sets	Average MSOP Score	Normalized Average MSOP Score	Execution Time (h)	Sequence Sets	Average MSOP Score	Normalized Average MSOP Score	Execution Time (h)
ClustalW	1aboA	0.714	-0.1798	--	1pii	**0.864**	**0.9797**	--
MAFFT		0.741	0.3395	--		0.764	0.6175	--
BW		0.6418	-1.5686	**7.3056e-004**		0.1647	-1.5533	0.0024
SPSO		0.6959	-0.528	0.0123		0.3548	-0.8647	0.2456
RDPSO		0.7558	0.6242	0.0132		0.6283	0.126	0.2478
RDPSO-DGS		**0.7916**	**1.3128**	0.0147		0.7853	0.6947	0.2536
ClustalW	1idy	0.705	0.1297	--	5ptp	**0.966**	**1.0986**	--
MAFFT		0.753	0.5113	--		0.840	0.1792	--
BW		0.5132	-1.3949	**6.9722e-004**		0.6053	-1.5332	0.0021
SPSO		0.5582	-1.0372	0.0116		0.7016	-0.8306	0.2230
RDPSO		0.7738	0.6766	0.0127		0.8548	0.2872	0.2314
RDPSO-DGS		**0.8289**	**1.1146**	0.0136		0.9249	0.7987	0.2367
ClustalW	451c	**0.7190**	**1.0039**	--	gal4	0.483	0.2024	--
MAFFT		0.684	0.7566	--		0.572	0.7769	--
BW		0.3989	-1.2584	0.0008		0.2017	-1.6134	0.0040
SPSO		0.4463	-0.9234	0.0146		0.3273	-0.8027	0.5100
RDPSO		0.5121	-0.4583	0.0152		0.5328	0.5238	0.5233
RDPSO-DGS		0.7014	0.8795	0.0160		**0.5931**	**0.9131**	0.5369
ClustalW	1krn	**1.000**	**0.7565**	--	1ajsA	0.5710	0.383	--
MAFFT		**1.000**	**0.7565**	--		0.592	0.536	--
BW		0.8182	-1.1617	0.0008		0.2864	-1.690	0.0032
SPSO		0.7988	-1.3663	0.0130		0.4220	-0.7023	0.2844
RDPSO		0.9528	0.2585	0.0139		0.5844	0.4806	0.3369
RDPSO-DGS		**1.0000**	**0.7565**	0.0143		**0.6547**	**0.9927**	0.3413
ClustalW	1bbt3	0.638	-0.3448	--	glg	**0.941**	**0.9081**	--
MAFFT		0.726	0.8259	--		0.894	0.5845	--
BW		0.5347	-1.719	0.0018		0.5691	-1.6523	0.0082
SPSO		0.6428	-0.2809	0.2173		0.6942	-0.791	2.1006
RDPSO		0.7152	0.6822	0.2226		0.8616	0.3614	2.1494
RDPSO-DGS		**0.7268**	**0.8365**	0.2309		0.8947	0.5893	2.1805
ClustalW	kinase	0.736	0.6910	--	1tag	**0.963**	**1.4325**	--
MAFFT		**0.836**	**1.1506**	--		0.769	-0.2413	--
BW		0.2268	-1.6493	0.0029		0.6453	-1.3085	0.0152
SPSO		0.4643	-0.5577	0.2652		0.7147	-0.7098	3.2882
RDPSO		0.5766	-0.0416	0.2714		0.7977	0.0063	3.3290
RDPSO-DGS		0.6742	0.407	0.2787		0.8921	0.8208	3.4174

for each sequence set, and the normalized average MSOP scores of each method for each sequence set. The MSOP score in equation (5) was used for HMM training in this case since the reference alignments, i.e., the prior knowledge of the resulting alignments, is available. The standard deviations of the MSOP scores are very small, so they are not listed in the table. The average execution time of each training method over 20 runs for each sequence set is also given in the table. Additionally, the MSOP scores of the alignments obtained by the ClustalW and MAFFT programs are also listed in the table. The normalized average MSOP score of each method for each sequence set was obtained in a similar way as in Tables 2, 3, and 4. For the laboA sequence set, the HMMs trained by the RDPSO-DGS and RDPSO yielded the alignments that had the best and the second best average MSOP scores and normalized average MSOP scores. The next best alignments were produced by the MAFFT and ClustalW programs, respectively. The same goes for the lidy sequence set. For the 451c sequence set, ClustalW and the HMM trained by the RDPSO-DGS yielded the best and second best alignment, respectively. The MFATT program performed the third best in the alignment of this sequence set. The next best-performing in MSA for this sequence data set is the RDPSO-trained HMM. For the lkrn sequence set, the MSOP scores of the alignments that were produced by ClustalW, MAFFT and the RDPSO-learned HMM were tied for the best MSOP scores. The next best MSOP score was that of the alignment generated by the RDPSO-trained HMM. For the lbbt3 sequence set, the RDPSO-DGS-trained HMM and MAFFT program aligned the sequence set with the best and second best MSOP scores, respectively. The alignment yielded by the RDPSO-learned HMM had the next best MSOP score. For the kinase sequence set, the MAFFT and ClustalW programs performed the best and the second best in MSA, as can be seen from the MSOP score or normalized MSOP scores of the obtained alignments. The MSOP score of the alignment resulted from the RDPSO-DGS-trained HMM was the next best one. For the 1pii sequence set, ClustalW aligned the sequences with the best MSOP score. The alignments with the second best and the third best MSOP scores were obtained by the RDPSO-DGS-learned HMM and MAFFT program. For the 5ptp sequence set, ClustalW, the RDPSO-DGS-trained HMM and RDPSO-trained HMM got the best, the second best and the third best alignments, respectively, in terms of MSOP scores. The HMM trained by the RDPSO-DGS produced the best alignment for the ga14 sequence set, and the MAFFT program generated the second best alignment for this sequence set, according to the MSOP scores or the normalized MSOP scores. For the 1ajsA sequence set, the alignment produced by the RDPSO-DGS-trained HMM had the best MSOP score. The second and third best alignments were those found by the MAFFT program and the RDPSO-learned HMM. The best alignment for the glg

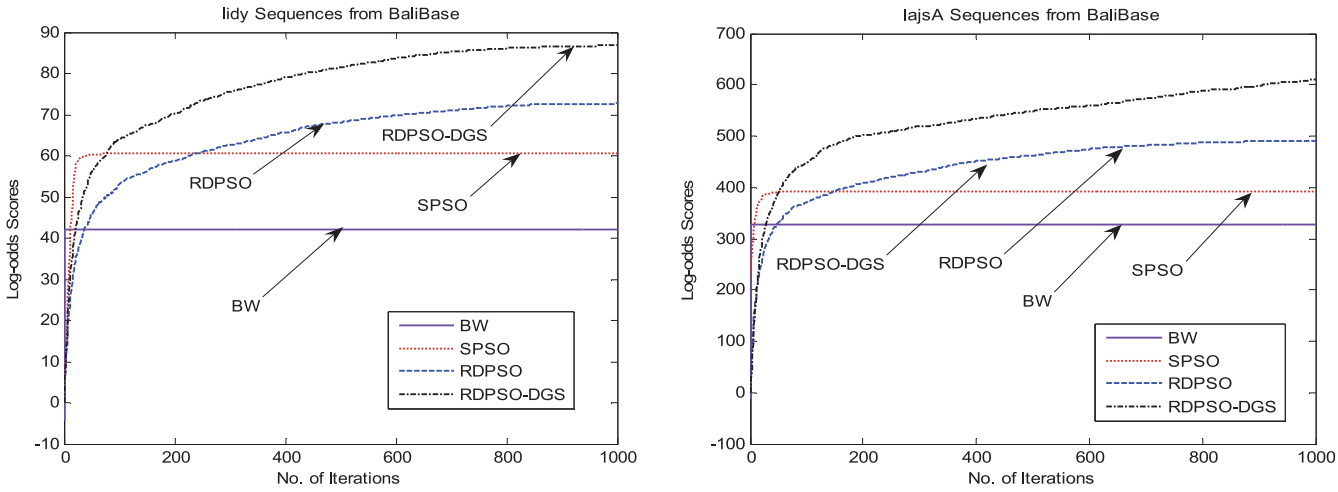

Fig. 2. Average log-odds scores found by the algorithms during training of the HMMs for lidy and lajsA sequences from the BAliBase database.

sequence set was obtained by the ClustalW program. The alignments produced by the RDPSO-DGS-trained HMM and MAFFT program were almost tied for the second best, since their MSOP scores are very similar. For the ltag sequence set, the first three best MSOP scores are those of the alignment produced by ClustalW, the RDPSO-DGS-learned HMM and RDPSO-learned HMM. The alignment yielded by the MAFFT program had the next best MSOP score. Among the training algorithms for HMMs, i.e., the RDPSO-DGS, RDPSO, SPSO, and BW, the HMMs trained by the RDPSO-DGS and RDPSO had the best and second best overall performance in the multiple sequence alignment for these sequence sets. As for the execution time of the HMM training, although the RDPSO-DGS, RDPSO and SPSO were more computationally consuming, the alignment accuracies obtained by the HMMs trained by them are much higher than the BW. As mentioned in the previous section, it is worth using these global search techniques and spending longer time to train HMMs for a more accurate multiple sequence alignment.

Fig. 2 shows the convergence process averaged over 20 runs of each learning algorithm on the course of the HMM learning process. It can be seen that the RDPSO-DGS had the best convergence properties among all the learning algorithms. In order to show the good performance of the RDPSO and RDPSO-DGS, we present in Fig. 3 the reference alignments of the lidy sequences and the alignments provided by the ClustalW program, the MAFFT program, and the HMMs learned by the BW, SPSO, RDPSO and RDPSO-DGS.

5.4 Discussion

In order to make an overall performance comparison among all the methods, we averaged the normalized average log-odds scores and the normalized average sum-of-pairs scores that are listed in Tables 2, 3, 4, and 5 for each method. Also, for the same purpose of overall performance comparison, all of the two types of normalized average scores for each method over all the tested

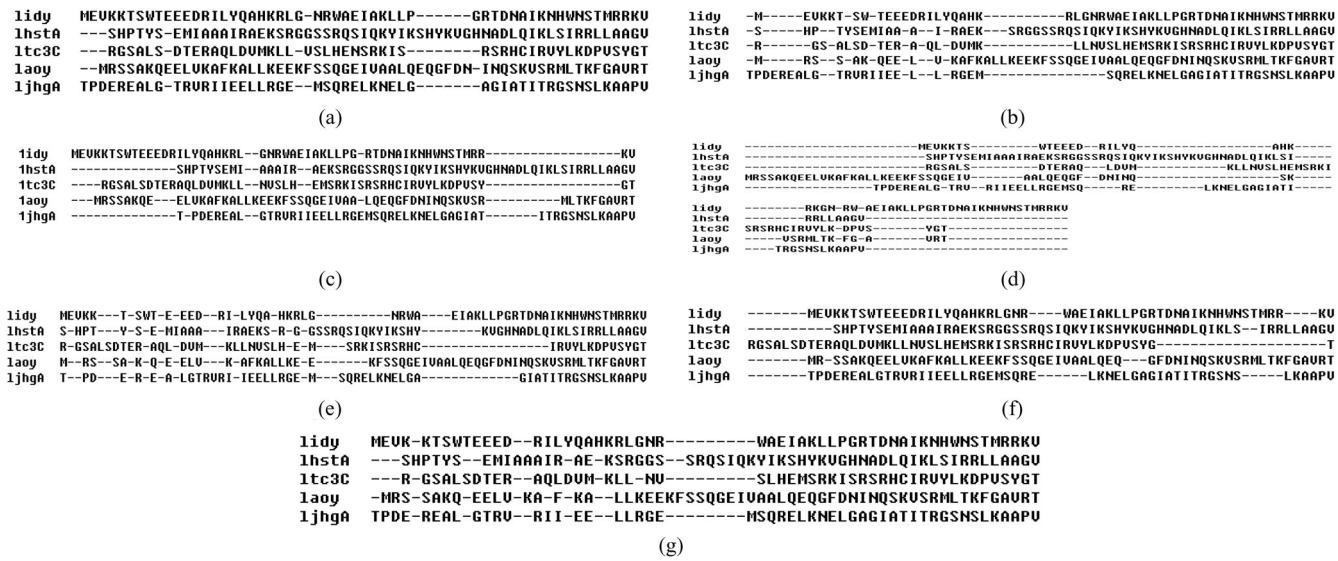

Fig. 3. (a) The reference alignment, and the resulting alignments for the lidy sequences generated by (b) ClustalW; (c) MAFFT; (d) HMM-BW; (e) HMM-SPSO; (f) HMM-RDPSO; (g) HMM-RDSPO-DGS.

TABLE 6
The Average Normalized Scores on all Data Sets

Algorithms	Average Normalized Log-odds Score	Average Normalized Sum-of-Pairs Score	Total Average Normalized Score
ClustalW	--	0.3942	0.3942
MAFFT	--	0.4862	0.4862
BW	-0.9841	-1.3792	-1.1816
SPSO	-0.5892	-0.7852	-0.6872
RDPSO	0.4001	0.3375	0.3688
RDPSO-DGS	1.1733	0.9464	1.0599

sequence sets were also averaged. This averaged normalized score is named total average normalized score. The results are presented in Table 6. Among all the training algorithms, including the RDPSO-DGS, RDPSO, SPSO and BW, both the average normalized log-odds score and sum-of-pairs score produced by the RDPSO-DGS-learned HMMs were better than those of any other competitor. This indicates that the RDPSO-DGS had the best overall performance in the HMM parameter learning for the MSA on the two benchmark data sets. Among all the training algorithms, the RDPSO algorithm showed to have the second best overall performance in HMM training, as its average normalized log-odds score, average normalized sum-of-pairs scores and total average normalized score were ranked the second among all the methods. The BW algorithm performed worst, among all the training algorithms, in HMM parameter learning tasks for the considered MSA problems.

Comparing all the tested multiple alignment methods, including ClustalW, MAFFT, the HMMs learned by the four training algorithms, the RDPSO-DGS-learned HMM had the best overall performance in MSA on the two data sets. The MAFFT program had the second best overall performance, as indicated by the averaged normalized sum-of-pairs score and the total average normalized score. The ClustalW program, the next best-performing method, slightly outperformed the RDPSO-trained HMMs in MSA for the given benchmark data sets. The comparison between these methods implies that an efficient learning algorithm is crucial for HMMs in order to show advantages in solving MSA problems.

It is well known that the parameter learning for a HMM is a time consuming task, particularly when meta-heuristic methods are used. As can be seen from Tables 2 3, 4, and 5 5, the BW consumed the least computational time since it is a local deterministic search technique. The RDPSO-DGS algorithm was slightly more time-consuming in learning the parameters of the HMMs for MSA than the RDPSO and SPSO, due to the computation of the diversity measure at each iteration during the learning process. However, it was worthwhile to consume tolerably more computational time in order to obtain the significant performance advantages of the RDPSO-DGS over its competitors.

In [21], the amino-acid data sets from the BAliBASE were tested. For each corresponding sequence set, it can be seen that the results of the RDPSO-DGS and RDPSO are better than those of the PSO variant in [21]. In [22], we tested the sequence sets from three data sets, two of which were the simulated nucleotide data sets and the amino-acid data sets from the BAliBASE we tested in this work. By comparing the results in this work and in [22], it can be concluded that the RDPSO-DGS outperformed all the methods proposed in [22]. Due to the space limitation in this paper, we have not listed the results of the methods in [21] and [22].

6 Conclusion

This paper focused on the HMM-based method for multiple sequence alignment. A novel PSO variant, the RDPSO algorithm along its improved version, named as the RDPSO-DGS, was proposed as a new optimization method and applied to train HMMs for MSA. The proposed RDPSO and RDPSO-DGS were used for the training of HMMs for MSA problems on two benchmark data sets, and compared with the popular ClustalW and MAFFT programs.

The proposed RDPSO algorithm and its variant were inspired by the free electron model in metal conductors placed in an external electric field. Analogous to the movement of electrons toward the location with minimum potential energy in the conductor, the particle in the RDPSO algorithm has both random (thermal) motion and drift motion, which implement the global search and the local search ability of the particle, respectively, in order to find the global minimum in the search space. Therefore, the velocity of the particle is the superposition of the random velocity and the drift velocity, which leads to a very different update equation for the particle's velocity. In order to improve the RDPSO even further, a diversity control strategy was incorporated into the algorithm and, thus, the RDPSO-DGS was proposed. With the diversity measure being controlled and maintained at a certain level, the stagnation of the particle swarm can be prevented effectively, and, consequently, the global search ability of the algorithm is enhanced considerably.

The experimental results showed that, with either the log-odds score or the sum-of-pairs score being the fitness value for the learning tasks, the RDPSO and RDPSO-DGS were able to produce the HMMs that had better scores than other learning algorithms, i.e., the SPSO and BW, for the simulated nucleotide data set and the amino-acid data set from the BaliBASE. Comparing with two popular MSA programs, ClustalW and MAFFT, the RDPSO-DGS-learned HMMs also showed a better overall performance than the two programs, as indicated by the total average normalized scores. In addition, the comparison of the average execution time among all the learning algorithms indicated that, in each experiment, the RDPSO-DGS and RDPSO algorithms consumed more CPU time for training the HMMs than the BW algorithm. However, considering its significant performance advantages over its competitors, we can conclude that the RDPSO-DGS and RDPSO is a very efficient approach for HMM parameter learning for MSA problems.

Acknowledgments

This work was supported by the Natural Science Foundation of China (NSFC) (Project Number: 601190117, 61105128, 61373055), by the Program for New Century Excellent Talents in University (Project Number: NCET-11-0660), by the RS-NSFC International Exchange Program (Project Number: 61311130141), by the Natural Science

Foundation of Jiangsu Province, China (Project Number: BK2010143), and by Key grant Project of Chinese Ministry of Education (Project Number: 311024).

REFERENCES

[1] D.W. Mount, *Bioinformatics: Sequence and Genome Analysis*. Cold Spring Harbor Laboratory Press, 2001.

[2] L. Wang and T. Jiang, "On the Complexity of Multiple Sequence Alignment," *J. Computational Biology*, vol. 1, no. 4, pp. 337-348, 1994.

[3] D.F. Feng and R.F. Doolittle, "Progressive Sequence Alignment as a Prerequisite to Correct Phylogenetic Trees," *J. Molecular Evolution*, vol. 25, no. 4, pp. 351-360, 1987.

[4] J.D. Thompson, D.G. Higgins, and T.J. Gibson, "CLUSTALW: Improving the Sensitivity of Progressive Multiple Sequence Alignment through Sequence Weighting, Position-Specific Gap Penalties and Weight Matrix Choice," *Nucleotide Acids Research*, vol. 22, no. 22, pp. 4673-4680, Nov. 1994.

[5] C. Notredame, D.G. Higgins, and J. Heringa, "T-Coffee: A Novel Method for Fast and Accurate Multiple Sequence Alignment," *J. Molecular Biology*, vol. 302, no. 1, pp. 205-217, Sept. 2000.

[6] C.B. Do, M.S. Mahabhashyam, M. Brudno, and S. Batzoglou, "ProbCons: Probabilistic Consistency-Based Multiple Sequence Alignment," *Genome Research*, vol. 15, no. 2, pp. 330-340, Feb. 2005.

[7] R.C. Edgar, "MUSCLE: Multiple Sequence Alignment with High Accuracy and High Throughput," *Nucleic Acids Research*, vol. 32, no. 5, pp. 1792-1797, Mar. 2004.

[8] K. Katoh and D.M. Standley, "MAFFT Multiple Sequence Alignment Software Version 7: Improvements in Performance and Usability," *Molecular Biology and Evolution*, vol. 30, no. 4, pp. 772-780, Jan. 2013.

[9] J. Kim, S. Pramanik, and M.J. Chung, "Multiple Sequence Alignment Using Simulated Annealing," *Bioinformatics*, vol. 10, no. 4, pp. 419-426, July 1994.

[10] A.V. Lukashin, J. Engelbrecht, and S. Brunak, "Multiple Alignment Using Simulated Annealing: Branch Point Definition in Human mRNA Splicing," *Nucleic Acids Research*, vol. 20, no. 10, pp. 2511-2516, May 1992.

[11] K. Chellapilla and G.B. Fogel, "Multiple Sequence Alignment Using Evolutionary Programming," *Proc. the First Congress on Evolution Composition*, vol. 1, pp. 445-452, 1999.

[12] C. Notredame and D.G. Higgins, "SAGA: Sequence Alignment by Genetic Algorithm," *Nucleic Acids Research*, vol. 24, no. 8, pp. 1515-1524, Apr. 1996.

[13] P. Baldi, Y. Chauvin, T. Hunkapiller, and M.A. McClure, "Hidden Markov Models of Biological Primary Sequence Information," *Proc. Nat'l Academy Sciences USA*, vol. 91, no. 3, pp. 1059-1063, Feb. 1994.

[14] K. Karplus, C. Barrett, and R. Hughey, "Hidden Markov Models for Detecting Remote Protein Homologies," *Bioinformatics*, vol. 14, no. 10, pp. 846-856, 1998.

[15] A. Krogh, M. Brown, I.S. Mian, K. Sjolander, and D. Haussler, "Hidden Markov Models in Computational Biology: Applications to Protein Modeling," *J. Molecular Biology*, vol. 235, no. 5, pp. 1501-1531, Feb. 1994.

[16] L.E. Baum, T. Petrie, G. Soules, and N. Weiss, "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains," *Annals of Math. Statistics*, vol. 41, no. 1, pp. 164-171, 1970.

[17] R. Hughey and A. Krogh, "Hidden Markov Models for Sequence Analysis: Extension and Analysis of the Basic Method," *Bioinformatics*, vol. 12, no. 2, pp. 95-107, 1996.

[18] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge Univ. Press, 1998.

[19] S.R. Eddy, "Multiple Alignment Using Hidden Markov Models," *Proc. the Int'l Conf. Intelligent Systems for Molecular Biology*, vol. 3, pp. 114-120, 1995.

[20] S. Kwong, C. Chau, K. Man, and K. Tang, "Optimisation of HMM Topology and Its Model Parameters by Genetic Algorithm," *Pattern Recognition*, vol. 34, no. 2, pp. 509-522, Feb. 2001.

[21] T.K. Rasmussen and T. Krink, "Improved Hidden Markov Model Training for Multiple Sequence Alignment by a Particle Swarm Optimization-Evolutionary Algorithm Hybrid," *BioSystems*, vol. 72, pp. 5-17, 2003.

[22] J. Sun, X. Wu, W. Fang, Y. Ding, H. Long, and W. Xu, "Multiple Sequence Alignment Using the Hidden Markov Model Trained by an Improved Quantum-Behaved Particle Swarm Optimization," *Information Sciences*, vol. 182, no. 1, pp. 93-114, Jan. 2012.

[23] F. van den Bergh, "An Analysis of Particle Swarm Optimizers," PhD dissertation, Univ. Pretoria, Pretoria, South Africa, 2002.

[24] J. Kennedy, "Some Issues and Practices for Particle Swarms," *Proc. IEEE Swarm Intelligence Symp.*, pp. 801-808, 2007.

[25] M.A. Omar, *Elementary Solid State Physics: Principles and Applications*. Addison-Wesley, 1993.

[26] L.R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257-285, Feb. 1989.

[27] S. Henikoff and J.G. Henikoff, "Amino Acid Substitution Matrices from Protein Blocks," *Proc. Nat'l Academy Sciences USA*, vol. 89, no. 22, pp. 10915-10919, Nov. 1992.

[28] J. Thompson, F. Plewniak, and O. Poch, "A Comprehensive Comparison of Multiple Sequence Alignment Programs," *Nucleotide Acids Research*, vol. 27, no. 13, pp. 2682-2690, 1999.

[29] J. Kennedy and R.C. Eberhart, "Particle Swarm Optimization," *Proc. IEEE Int'l Conf. Neural Networks*, pp. 1942-1948, 1995.

[30] Y. Shi and R.C. Eberhart, "A Modified Particle Swarm Optimizer," *Proc. IEEE Int'l Conf. Evolutionary Computation*, pp. 69-73, 1998.

[31] M. Clerc, "The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization," *Proc. Congress on Evolutionary Computation*, pp. 1951-1957, 1999.

[32] D. Bratton and J. Kennedy, "Defining a Standard for Particle Swarm Optimization,," *Proc. IEEE Swarm Intelligence Symp.*, pp. 120-127, 2007.

[33] J. Sun, W Fang, X. Wu, V. Palade, and W. Xu, "Quantum-Behaved Particle Swarm Optimization: Analysis of the Individual Particle's Behavior and Parameter Selection," *Evolutionary Computation*, vol. 20, no. 3, pp. 349-393, Aug. 2012.

[34] J. Sun, X. Wu, V. Palade, W. Fang, C.-H. Lai, and W. Xu, "Convergence Analysis and Improvements of Quantum-Behaved Particle Swarm Optimization," *Information Sciences*, vol. 193, pp. 81-103, 2012.

[35] J. Sun, W. Fang, V. Palade, X. Wu, and W. Xu, "Quantum-Behaved Particle Swarm Optimization with Gaussian Distributed Local Attractor Point," *Appl. Math. & Comput.*, vol. 218, no. 7, pp. 3763-3755, 2011.

[36] M. Clerc and J. Kennedy, "The Particle Swarm-Explosion, Stability and Convergence in a Multidimensional Complex Space," *IEEE Trans. Evolutionary Computation*, vol. 6, no. 2, pp. 58-73, Feb. 2002.

[37] R.K. Ursem, "Diversity-Guided Evolutionary Algorithms," *Proc. the Seventh Parallel Problem Solving from Nature Conf.*, pp. 462-471, 2002.

[38] J. Riget and J.S. Vesterstrøm, "A Diversity-Guided Particle Swarm Optimizer-the ARPSO," technical report, Univ. of Aarhus, Denmark, 2002.

[39] J. Stoye, D. Evers, and F. Meyer, "Rose: Generating Sequence Families," *Bioinformatics*, vol. 14, no. 2, pp. 157-163, 1998.

[40] T.H. Jukes and C.R. Cantor, "Evolution of Protein Molecules," in H.N. Munro, (ed.), *Mammalian Protein Metabolism*, Academic Press, New York, vol. 3, pp. 21-132, 1969.

Jun Sun received the PhD degree in control theory and engineering, and the MSc degree in computer science and technology from Jiangnan University, China, in 2009 and 2003, respectively. He is currently an associate professor with the Department of Computer Science and Technology, Jiangnan University, China. He is also a researcher at the Key Laboratory of Advanced Process Control for Light Industry (Ministry of Education), China. His major research areas and work are related to machine learning, computational intelligence, and bioinformatics, among others. He published more than 100 papers in journals, conference proceedings, and several books in the above areas.

Vasile Palade received the PhD degree from the University of Galati, Romania, in 1999. He is currently currently a Reader at Coventry University, United Kingdom, and he previously worked as a lecturer with the Department of Computer Science, University of Oxford, United Kingdom. His research interests are in the area of computational intelligence with application to bioinformatics, fault diagnosis, web usage mining, among others. He published more than 100 papers in journals and conference proceedings as well as several books. He is a senior member of the IEEE.

Xiaojun Wu received the BS degree in mathematics from the Nanjing Normal University, Nanjing, PR China, in 1991, the MS degree in 1996, and the PhD degree in pattern recognition and intelligent systems in 2002, both from the Nanjing University of Science and Technology, Nanjing, PR China, respectively. He joined Jiangnan University in 2006, where he is a professor. He has published more than 150 papers in his fields of research. He was a visiting researcher in the Centre for Vision, Speech and Signal Processing (CVSSP), University of Surrey, United Kingdom, from 2003 to 2004. His current research interests include pattern recognition, computer vision, fuzzy systems, neural networks, and intelligent systems.

Wei Fang received the PhD degree in information technology and engineering from Jiangnan University in 2008. He is an associate professor of computer science at the Jiangnan University. He is currently a visiting scholar at the University of Birmingham, United Kingdom. He has published more than 20 scientific papers in journals and international conferences. His current research interests involve the design and application of evolutionary algorithms, especially swarm intelligence algorithms.