A PROBABILISTIC CODING BASED QUANTUM GENETIC ALGORITHM FOR MULTIPLE SEQUENCE ALIGNMENT

Hongwei Huo*, Qiaoluan Xie, and Xubang Shen

School of Computer Science and Technology, Xidian University
Xi'an, Shaanxi 710071, P.R.China
**Email: hwhuo@mail.xidian.edu.cn*

Vojislav Stojkovic

Computer Science Department, Morgan State University
Baltimore, Maryland 21251, USA
Email: vojislav.stojkovic@morgan.edu

This paper presents an original Quantum Genetic algorithm for Multiple sequence ALIGNment (QGMALIGN) that combines a genetic algorithm and a quantum algorithm. A quantum probabilistic coding is designed for representing the multiple sequence alignment. A quantum rotation gate as a mutation operator is used to guide the quantum state evolution. Six genetic operators are designed on the coding basis to improve the solution during the evolutionary process. The features of implicit parallelism and state superposition in quantum mechanics and the global search capability of the genetic algorithm are exploited to get efficient computation. A set of well known test cases from BAliBASE2.0 is used as reference to evaluate the efficiency of the QGMALIGN optimization. The QGMALIGN results have been compared with the most popular methods (CLUSTALX, SAGA, DIALIGN, SB_PIMA, and QGMALIGN) results. The QGMALIGN results show that QGMALIGN performs well on the presenting biological data. The addition of genetic operators to the quantum algorithm lowers the cost of overall running time.

1. INTRODUCTION

Multiple Sequence Alignment (MSA) is one of the most challenging tasks in bioinformatics. Most of the MSA methods are based on the dynamic programming approach. The dynamic programming approach requires time proportional to the product of the lengths of sequences which makes it computationally difficult. In the general case, the theoretical sound and biologically motivated scoring methods are not straightforward connected. Usually, it is hard to efficiently align more than a few sequences. For larger instances, a variety of heuristics strategies have been developed. In general, two basic classes of MSA methods have been proposed: progressive alignment and iterative alignment[1].

Progressive alignment methods use dynamic programming to build MSA. The best known software system based on progressive alignment method is maybe CLUSTALW[2]. Other well-known MSA systems based on progressive alignment method are MULTALIGN[3], T-COFFEE[4], MAFFT[5], MUSCLE[6], Align-m[7], and PROBCONS[8]. Mostly, they target proteins or short DNA sequences. The main advantages of progressive alignment methods are speed and simplicity. The main disadvantage of progressive alignment methods is that mistakes in the initial alignments of the most closely related sequences are propagated to the multiple alignments.

Iterative alignment methods depend on algorithm that produces an alignment and refines it through a serious of iterations until no more improvement can be made. Iterative alignment methods can be deterministic or stochastic. The deterministic iterative strategies involve extracting sequences one by one from a multiple alignment and realigning them to the remaining sequences. Stochastic iterative alignment methods include Hidden Markov Model (HMM) training, simulated annealing[9] and evolutionary computation[10]. The main advantage of stochastic iterative alignment methods is a good separation between the optimization process and evaluation criteria. The main disadvantages of stochastic iterative alignment methods are local optima, slow convergent speed, and lacking a specific termination condition.

* Corresponding author.

In the last twenty years a growing interest in quantum computation and quantum information is due to the possibility to efficiently solve hard problems for conventional computer science paradigms. Quantum algorithms exploit the laws of quantum mechanics. The quantum computation can dramatically improve performance for solving problems like factoring and search in an unstructured database. Genetic algorithms are stochastic search algorithms based on the principles of natural selection and natural genetics. They work on a set of chromosomes, called population that evolves by means of crossover and mutation towards a maximum of the fitness function. Genetic algorithms are efficient and flexible algorithms.

Han-Kim[11] proposed the possibility to integrate the quantum and genetic algorithms. Huo and Stojkovic[12] presented Quantum-inspired Evolutionary Algorithms (QEA) with a quantum representation. By adapting a qubit chromosome representation, a quantum population is generated. Classical population is generated by performing measurements on the quantum population. The best elements are searched in the classical population and used to update the quantum population. Experiments are carried out on the knapsack problem.

Now we go one step further. We redesigned QEA to solve the multiple sequence alignment problem. This paper presents a Quantum Genetic algorithm for Multiple sequence ALIGNment (QGMALIGN). It exploits the expression power of quantum mechanics in the coding and shows how to take advantage of quantum phenomena to efficiently speed up classical computation. A new probabilistic coding method for the MSA representation is given. A quantum rotation gate as a mutation operator is used to guide the quantum state evolution of the population. Six genetic operators are designed on the basis of the coding to help to improve the solutions during the evolutionary process. The features of implicit parallelism and state superposition in quantum mechanics and the global search capability of the genetic algorithm are exploited to perform efficient computation. The COFFEE (Consistency based Objective Function For alignmEnt Evaluation)[13] function is used to measure individual fitness. To demonstrate QGMALIGN's effectiveness, a set of well known test cases from BAliBASE2.0 is used as reference to evaluate the efficiency of the optimization for QGMALIGN. The QGMALIGN results have been compared with the most popular methods (CLUSTALX, SAGA, DIALIGN, SB_PIMA, and QGMALIGN) results. The QGMALIGN results show that QGMALIGN performs well on the presenting biological data.

2. CODING AND FITNESS EVALUATION

2.1. Quantum probabilistic coding

The basic information unit of quantum computation is the qubit. A qubit is a two-level quantum system and can be considered a superposition of two independent basis states $|0\rangle$ and $|1\rangle$, denoted by:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle. \quad (1)$$

where α and β are complex number such that $|\alpha|^2 + |\beta|^2 = 1$.

A two-level classical system can be only in one of the basis states $|0\rangle$ or $|1\rangle$. α and β are probability amplitudes associated with the $|0\rangle$ state and the $|1\rangle$ state, respectively. If we want to transfer information from the quantum system to a classical 0-1 system, we have to perform measurement of the quantum state, whose result is probabilistic: we get the state $|0\rangle$ with probability $|\alpha|^2$ and the state $|1\rangle$ with probability $|\beta|^2$. There is no way to know exactly both values. We cannot clone the unknown state $|\psi\rangle$ as stated by the No cloning theorem.

The evolution of a quantum system is described by a special linear operator, unitary operator U_f, which operates on qubits.

$$U_f|\psi\rangle = U_f[\alpha|0\rangle + \beta|1\rangle] = \alpha U_f|0\rangle + \beta U_f|1\rangle$$

An important consequence of the linearity of quantum operators is that the evolution of a two-level quantum system is the linear combination of the evolution of the basis states $|0\rangle$ and $|1\rangle$. It is possible to compute $f(x)$ for many different values of x simultaneously in a single application of U_f.

A system of m-qubits can represent 2^m different states simultaneously. The observing quantum state collapses to a single state among these states.

A qubit individual in a quantum genetic algorithm is defined as follows:

$$q = \begin{bmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_m \\ \beta_1 & \beta_2 & \dots & \beta_m \end{bmatrix}. \quad (2)$$

where $|\alpha_i|^2 + |\beta_i|^2 = 1$, $i = 1, 2, \ldots, m$. The quantum coding is inspired by the features of quantum mechanics. During the evolution process of a quantum system, we need to compute $|\alpha_i|^2$ to obtain the probability matrix of the quantum system and then to transform it to the corresponding binary matrix by performing the quantum observation. The quantum variation has an indirect effect on the qubit state by changing the values of α_i at the expense of some extra space for storing probability matrix. It is disadvantageous for solving complex problems. The new quantum probabilistic coding is proposed for representing the multiple sequence alignment. This way of coding shields the underlying information of complex α and β. The genetic operators can perform directly on the probabilistic matrix while the feature of superposition from quantum mechanics is preserved.

Assume that $Q(t) = \{q^t_1, q^t_2, \ldots, q^t_n\}$ is a population of the generation t, where n is the number of chromosomes in the population. The chromosome q^t_j is defined as

$$q^t_j = [p^t_{j1} \mid \; p^t_{j2} \mid \; p^t_{j3} \mid \; \ldots \; \mid p^t_{jm}]. \qquad (3)$$

where $p^t_{ji} = |\beta^t_{ji}|^2$, p^t_{ji} is the probability of the letter being observed with value one at that position, p^t_{ji} is the length of the chromosome. When $p^t_{ji} = 1/2$, there are 2^m underlying different linear superposition states occurring with the same probability. The probabilistic coding that substitutes the form of (2) simplifies the encoding and saves the running time of the algorithms while maintaining the quantum properties.

2.2. Mapping the coding to the solution to MSA

The MSA problem can be formulated mathematically as follows: Given n sequences $S = \{S_1, S_2, \ldots, S_n\}$ defined over the finite alphabet Σ, where $n \geq 2$. S_{ij} where $1 \leq i \leq n$, $1 \leq j \leq l_i$ is a character of the alphabet Σ, where l_i is the length of S_i. A potential alignment is the set $S' = \{S'_1, S'_2, \ldots, S'_n\}$, satisfying the following conditions: (i) The sequence S'_i is the extension of S_i and is defined over the alphabet $\Sigma' = \Sigma \cup \{-\}$. '-' denotes a gap. The deletion of gaps from S'_i gives S_i; (ii) S'_i and S'_j have the same length; (iii) An objective function is a reference to biological significance that evaluates the quality of alignments.

An alignment for MSA can be obtained by measuring the quantum probabilistic matrix. The system collapses to a superposition of states that have the observed fitness. The measurement operation stems from quantum observation on a quantum computer. The difference is that the quantum observation on a quantum state can be performed many times without destroying all other configurations as it is not done in pure quantum systems. The quantum observation allows us to extract one state from the superposition of quantum probabilistic representation, having value of one with probability p_{ji} and zero with probability $1-p_{ji}$. The result of this operation is a binary matrix (BM, see Fig. 1). '1'means that there is a letter at the position of the original sequence. '0' means a gap. The number of '1s' in a row has to be the length of the sequence. The result must be repaired to fit the length of the sequence. Fig. 2 shows the alignment obtained from the binary matrix.

$$\begin{bmatrix} (0.85 & 0.34 & 0.85 & 0.95) \\ (0.77 & 0.50 & 0.87 & 0.45) \\ (0.85 & 0.42 & 0.90 & 0.65) \end{bmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

Fig. 1. Measurement operation.

$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} A & \_ & F & R \\ A & \_ & F & \_ \\ A & E & F & R \end{pmatrix}$$

Fig. 2. From binary matrix to an alignment.

2.3. Objective function

Objective function is used to measure the quality of MSA, which provides the basis for selection mechanism of the algorithm. Ideally, what score is better, then the multiple alignment is more biologically relevant. In this paper, we have used the COFFEE function as fitness criterion. Firstly we have a set of pairwise reference alignments (library), which includes $n*(n-1)/2$ pairwise alignments. The COFFEE function evaluates the consistency between the current multiple alignment and the pairwise alignments contained in the library. It can be formalized as follows:

$$COST(A) = \left[\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} W_{ij} \times SCORE(A_{ij})\right] \Big/ \left[\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} W_{ij} \times LEN(A_{ij})\right] \quad (4)$$

where N is the number of sequences to be aligned. A_{ij} is the pairwise projection (obtained from the multiple alignment) of sequences S_i and S_j, $LEN(A_{ij})$ is the length of this alignment, $SCORE(A_{ij})$ is the number of aligned pairs of residues that are shared between A_{ij} and the corresponding pairwise alignment in the library, and W_{ij} is the weight associated with the pairwise alignment.

3. THE OPTIMIZATION MECHANISM OF QUANTUM GENETIC ALGORITHM

3.1. Quantum mutation

The mutation operator in standard genetic algorithms is performed randomly. Individual variation of the evolutionary process with random disturbances can slow the convergent process. Quantum evolutionary processes are unitary transformations: rotations of complex space. Repeated application of a quantum transform may rotate the state closer and closer to the desired state. The basic result for quantum evolutionary process is that an unitary matrix can be represented by a finite set of universal gates. The quantum state evolution is guided by adding the optimal individual information to the variation so as to increase the probability of some quantum states to observe the better alignments and improve the convergence for the algorithm. The quantum rotation gate is the quantum unitary transformation U, defined as follows:

$$U(\delta\theta) = \begin{bmatrix} \cos(\delta\theta) & -\sin(\delta\theta) \\ \sin(\delta\theta) & \cos(\delta\theta) \end{bmatrix}$$

the angles $\delta\theta$ can be found in table 1. The rotation gate is used to update the quantum state.

$$\begin{bmatrix} \alpha_i' \\ \beta_i' \end{bmatrix} = U(\delta\theta) \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} = \begin{bmatrix} \cos(\delta\theta) & -\sin(\delta\theta) \\ \sin(\delta\theta) & \cos(\delta\theta) \end{bmatrix} \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix}. \quad (5)$$

The quantum rotation gate is implemented by rotating the complex space. In Fig. 3, $|\alpha|^2$ gives the probability that the qubit will be found in the $|0\rangle$ state and $|\beta|^2$ gives the probability that the qubit will be found in the $|1\rangle$ state. Counterclockwise rotation in the first and third quadrants will increase the probability amplitude $|\beta|^2$, while in the second and fourth quadrants will increase the probability amplitude $|\alpha|^2$.

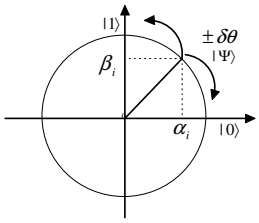


Fig. 3. Quantum observation: a projection on the basis.

According to the quantum probability coding, expression (5) can be simplified as follows:

$$p_i' = \cos^2(\delta\theta)p_i + \sin^2(\delta\theta)(1-p_i) + 2\cos(\delta\theta)\sin(\delta\theta)\sqrt{p_i(1-p_i)}\,. \quad (6)$$

Eq. (6) hides the influence of the sign of α_i and β_i on p_i. The unitary transformation makes p_i to take real values between 0 and 1. Only angles in the first quadrant can be took into account, as shown in Fig. 2.

The setting of rotation angle $\delta\theta$ is through experimentation and refer to the results in reference 11. The settings of $\delta\theta$ are application dependent. Many factors have an influence on the selection of the rotation angles, including the numbers of iterations associated with the characteristics of the sequences, the diversity of the population and convergent rate. Following the experimentation on the multiple sequence alignment problem, a lookup table for the choice of $\delta\theta$ is shown in the table 1. The values of the fitness for the best chromosome in the third column in the table have all values false because when the genetic operators perform on the chromosomes - the best one has been optimal in the current population.

Table 1. Lookup table of the rotation angle $\delta\theta$.

x_i	$best_i$	$f(x) \le f(best)$	$\delta\theta$
0	0	false	-0.005π
0	1	false	0.025π
1	0	false	-0.025π
1	1	false	0.005π

3.2. Genetic operators

The quantum mutation operator can bring good diversity of population. However, for the complexity of MSA, it is more probably for the evolutionary process to trap into the local optimum. Therefore, several

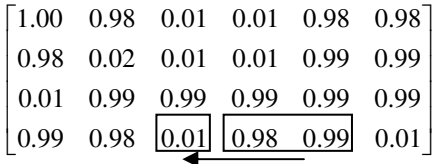
genetic operators are designed to avoid local optimum inspired by SAGA[10], which enhanced the capabilities to find the global optimal solutions.

3.2.1. *Local adjustment mutation operators*

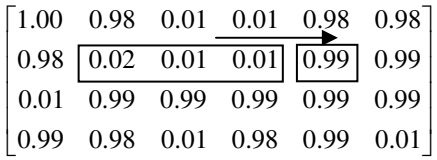
To improve the convergence - the better evolutionary strategies are needed. Inserting a gap to the left or to the right of the same position in each of the selected sequences often generate a better arrangement. An operator is designed to move blocks of residues or gaps inside an alignment. Two local adjustment operators are designed: the ResidueBlockShuffle operator and the GapBlockShuffle operator.

ResidueBlockShuffle: Move a full block without gaps to the right or to the left one position. A gap is inserted into the left or the right to that position. The block of randomly chosen length is chosen at a random position. Fig. 4(a) outlines this mechanism.

GapBlockShuffle: Split the block horizontally with the probability 15% and move one of the sub-blocks to the left or to the right. Move a full block of gaps with the probability 85% to the right or left until it merges with the next block of gaps, as Fig. 4(b) indicates.



(a) ResidueBlockShuffle



(b) GapBlockShuffle

Fig. 4. Local adjustment mutation operator examples.

3.2.2. *Global mutation operators*

BlockMove: Find a block with gaps randomly in an alignment, with width between two and length of the sequence and exchange position of the block with the position of a non-gap block. A special treatment for the gap-column. Fig. 5 shows how the BlockMove operator works. The length of the migration block is generated at random. The new location is taken from the nearby position including non-gaps with a large probability and randomly generated. Migrates to the neighbor with a large probability. The operator has a good effect on avoiding the local optimum.

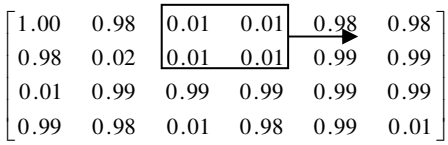
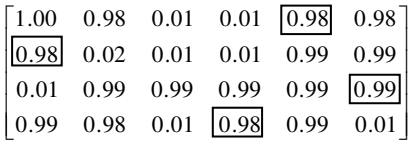
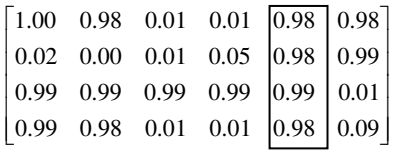


Fig. 5. Global mutation operator example: BlockMove.

ConsistencyShuffle: To make full use of the information from pairwise alignment library to perform the corresponding positions of adjustment and alignment, the ConsistencyShuffle operator, inspired by PHGA-COFFE, is designed to adjust the relative position of the residues. The process is as follows: Find a non-gap location of a sequence randomly in the multiple alignment, such as the positions with box in Fig. 6(a); Find the relative positions at which the selected sequence is aligned in the pairwise alignments library and record them in an array; Adjust gaps in the alignment so that the letters of the site for the multiple alignment coincide with the corresponding ones in pairwise alignment library, see Fig. 6(b).



(a) Finding the position in the pairwise library



(b) Adjustment

Fig. 6. ConsistencyShuffle.

The Crossover operators merge two different alignments with a higher quality into a new one. QGMALIGN implemented two different types of crossover: SingleCrossover and UniformCrossover. The former may be very disruptive. To avoid this drawback, the UniformCrossover operator is designed to promote multiple exchanges between two parents in a more subtle manner. Exchanges are promoted between zones of homology. In QGMALIGN, check whether or not the two chromosomes can do UniformCrossover, otherwise do SingleCrossover.

SingleCrossover: The X-shaped crossover is performed at the point where the perfected matched column belongs to, as shown in Fig. 7. After the crossover, the two new alignments maybe don't satisfy the constraints to the length of the sequence. The new chromosomes and the original chromosomes have different number of gaps. So we have to adjust the new chromosomes. We change p_{ij} with 1- p_{ij} in the shadowed area at random until the requirement for the number of gaps is met.

UniformCrossover: Find the crossover position in the two selected alignments, respectively. Children are produced by swapping blocks between the two parents where each block is randomly chosen between two positions. The shadowed blocks (see Fig. 8) are different areas between the two new alignments, coming from the two parents. During the process, the gaps are adjusting at random and the strategies are the same as the ones used in SingleCrossover. The choice of crossover points must satisfy the constraints: (i) The distance between the crossover positions is at least ten; (ii) At least one of the points is not available in another alignment.

3.2.3. *The Selection operator*

The Selection operator chooses the good alignments with a probability based on their fitness measured by OF(Objective Funcation). The selection operator makes sure that the good alignments survive and an optimal alignment can be found. It acts the same roles as the process of migration in evolutionary algorithms. The

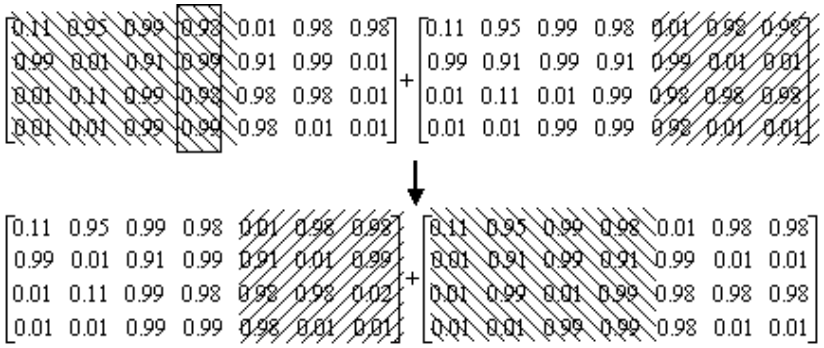


Fig. 7. SingleCrossover operator.

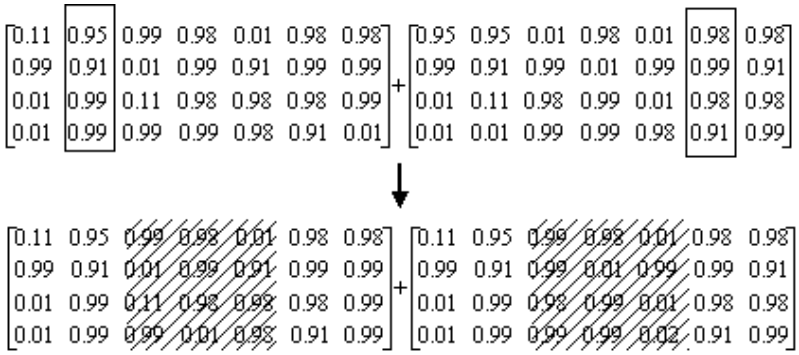


Fig. 8. UniformCrossover operator.

selection mechanisms in QGMALIGN are: typically 30% of the new generation is directly from the previous generation with the fittest alignments and the remaining 70% of the chromosomes in the new generation are created by roulette wheel selection.

3.3. Building the pairwise alignment library

In the QGMALIGN, the Needleman-Wunsch algorithm is used to build the pairwise alignment library and $n*(n-1)/2$ pairwise alignments are obtained. The BLOSUM matrices are chosen as the substitute matrix for protein sequences. The BLOSUM series ranges from BLOSUM30 to BLOSUM90, which one to choose depends on the distance between the two sequences, that is, the similarity of the two sequences. The penalty function is defined as follows:

$$penalty(gaps) = GOP + NG*GEP \quad (7)$$

where *GOP* (Gap Open Penalty) is a penalty for opening a new gap, *GEP* (Gap Extension Penalty) is a penalty for extending the length of an existing gap, and *NG* is the length of the gaps after the extension.

4. ALGORITHM

To perform multiple sequence alignment, the MSA method QGMALIGN is presenetd. QGMALIGN is derived from applying genetic algorithm in quantum computation. It uses a m-qubit representation variation of the form (3). For each representation, a binary matrix is defined, where each entry is selected using the corresponding qubit probability, $|\alpha_i|^2$ or $|\beta_i|^2$. It follows that if $|\alpha_i|^2$ or $|\beta_i|^2$ approaches to 1 or 0, then the qubit chromosome converges to a single state and the diversity given by the superposition of states disappears gradually.

The quantum-inspired computing algorithm QGMALIGN can be summarized in four steps:

(i) Initialize the population $Q(t) = \{q^t_1, q^t_2,\ldots,q^t_n\}$ of n-qubit chromosomes, where

$$q^t_j = [p^t_{j1}\rangle \quad p^t_{j2}\rangle \quad p^t_{j3}\rangle \quad \ldots \quad |p^t_{jm}\rangle] \ , j = 1,\ldots,n;$$

(ii) Apply Hadamard gate to chromosome of the population and generates a superposition of all 2^n possible states;

(iii) A sequence of rotate gate and genetic operators to evolve the population;

(iv) Quantum measurements and evaluation.

4.1. QGMALIGN algorithm

The QGMALIGN can be presented as a pseudocode as follows:

Algorithm QGMALIGN

1. Build pairwise library.
2. Initial population QM of 10 chromosomes.
3. Measurement from QM to BMs.
4. Evaluate the solutions of BMs and save the best one to Best_BM.
5. **while** (not termination-condition) **do**
    6. Apply global mutation with a probability.
    7. Apply local mutation with a probability.
    8. Apply the quantum mutation according to the best solution Best_BM.
    9. Measurement from QM to BMs.
    10. **for** each BM_i **do** evaluate the corresponding alignment
    11. **if** (fitness of Best_BM < fitness of BMbest)
        12. Best_BM = BMbest
        13. Elite selection.

The procedure QGMALIGN works as follows. Line 1 uses the Needleman-Wunsch algorithm to build the pairwise alignment library with $n*(n-1)/2$ pairwise alignments. Line 2 initializes the population QM to 10 chromosomes. Line 3 extracts one state from the superposition of quantum probabilistic representation, having value of one with probability p_{ji} and zero with probability $1–p_{ji}$. The result of this operation is a binary matrix. ‘1’ means that there is a letter at the position of the original sequence. ‘0’means a gap. Line 4 uses the COFFEE function to evaluate the alignment and saves the current best alignment. Lines 2-13 refine an alignment through a serious of optimization mechanisms. A quantum rotation gate as a mutation operator is used to guide the quantum state evolution. Six genetic operators are designed on the coding basis to improve the solution during the evolutionary process. The procedure ternimates when the current best alignment is not improved after 1000 times iterations.

5. THE EXPERIMENTAL RESULTS AND ANALYSIS

5.1. The experimental results

The parameters in QGMALIGN have been set as follows: *GOP* = 5, *GEP* = 0.1, the size of population is 10, and *Tmax* = 30000. The probabilities for various operators are given in table 2.

The experimental database comes from benchmark BAliBASE2.0. SPS (Sum-of-Pairs Score), is used to evaluate the final alignment. Comparisons (see tables 3~7) of the experimental results have been made with the the most popular methods (CLUSTALX, SAGA, DIALIGN, SB_PIMA, and QGMALIGN). Experimental results show that QGMALIGN performs well.

Table 2. The probabilities for five genetic operators.

Name	Probability
ResidueBlockShuffle	0.36
GapBlockShuffle	0.36
BlockMove	0.25
ConsistencyShuffle	0.65
Crossover	0.15

Table 3. SPS of Ref1.

dataset	identity	Seq_no×length	CLUSTAL X	SAGA	DIALIGN	SB_PIMA	QGMALIGN
1idy	14%	5×65	0.705	0.342	0.018	0.145	0.450
1r69	13%	4×80	0.481	0.550	0.406	0.681	0.467
2trx	17%	4×95	0.754	0.801	0.728	0.451	0.515
1havA	15%	5×200	0.446	0.411	0.130	0.300	0.200
2hsdA	19%	4×260	0.691	0.771	0.679	0.470	0.313
Kinase	20%	5×280	0.736	0.862	0.764	0.733	0.345
1lvl	19%	4×450	0.632	0.619	0.699	0.559	0.223
1hfh	31%	5×125	0.917	0.945	0.410	0.868	0.687
1hpi	33%	4×75	0.861	0.916	0.785	0.909	0.762
1pfc	28%	5×110	0.988	0.994	0.894	0.927	0.808
451c	27%	5×85	0.719	0.662	0.729	0.541	0.554
1aym3	32%	4×235	0.969	0.955	0.962	0.976	0.720
1pii	32%	4×255	0.864	0.896	0.890	0.832	0.575
1pkm	34%	4×440	0.921	0.955	0.927	0.907	0.717
1csp	51%	5×70	0.993	0.993	0.980	1.000	0.921
1dox	46%	4×105	0.919	0.879	0.859	0.868	0.835
1fmb	49%	4×105	0.981	0.979	0.959	0.952	0.901
1plc	46%	5×95	0.958	0.931	0.931	0.904	0.931
2fxb	51%	5×65	0.945	0.951	0.945	0.945	0.946
9rnt	57%	5×105	0.974	0.965	0.864	0.970	0.978
1led	43%	4×250	0.946	0.923	0.516	0.987	0.765
1ppn	46%	5×230	0.989	0.983	0.648	0.962	0.910
1thm	49%	4×280	0.961	0.956	0.946	0.971	0.809
5ptp	43%	5×250	0.966	0.940	0.888	0.966	0.694
1gtr	42%	5×430	0.986	0.995	0.961	0.960	0.755
1rthA	42%	5×540	0.977	0.960	0.958	0.962	0.786

Table 4. SPS of Ref2.

dataset	identity	Seq_no×length	CLUSTALX	SAGA	DIALIGN	SB_PIMA	QGMALIGN
1idy	28%	19×65	0.515	0.548	F	F	0.920
1ubi	32%	15×100	0.482	0.492	F	0.129	0.763
1aboA	28%	15×85	0.650	0.489	0.384	0.391	0.573
1csy	29%	19×100	0.154	0.154	F	F	0.684
1r69	26%	20×80	0.675	0.475	0.675	0.675	0.738
1tvxA	34%	16×70	0.552	0.448	F	0.241	0.832
1tgxA	35%	19×80	0.727	0.773	0.630	0.678	0.697
2trx	34%	19×95	0.870	0.870	0.734	0.850	0.883
1sbp	23%	16×280	0.231	0.217	0.374	0.043	0.364
2hsdA	28%	20×250	0.484	0.498	0.262	0.039	0.601
1ajsA	35%	18×390	0.324	0.311	F	F	0.612
1pamA	35%	18×500	0.761	0.623	0.576	0.393	0.572
2myr	32%	17×490	0.904	0.825	0.840	0.727	0.736
4enl	48%	17×450	0.375	0.739	0.122	0.096	0.655

Table 5. SPS of Ref3.

dataset	identity	Seq_no×length	CLUSTALX	SAGA	DIALIGN	SB_PIMA	QGMALIGN
1idy	20%	27×70	0.273	0.364	F	F	0.468
1r69	19%	23×85	0.524	0.524	0.524	F	0.247
1ubi	20%	22×105	0.146	0.585	F	F	0.321
1pamA	34%	19×530	0.678	0.579	0.683	0.546	0.526
1ped	32%	21×425	0.627	0.646	0.641	0.450	0.372
1wit	22%	19×110	0.565	0.484	0.500	0.645	0.548
2myr	24%	21×540	0.538	0.494	0.272	0.278	0.547
4enl	41%	19×480	0.547	0.672	0.050	0.393	0.394

Table 6. SPS of Ref4.

dataset	identity	Seq_no×length	CLUSTAL X	SAGA	DIALIGN	SB_PIMA	QGMALIGN
1csp	32%	6×700	F	F	0.889	F	0.304
1vln	43%	14×230	0.879	0.606	0.545	0.636	0.372
1ckaA	26%	10×820	F	0.375	1.000	1.000	0.120
1mfa	18%	8×480	1.000	0.385	1.000	0.846	0.345
1ycc	36%	9×210	0.485	0.485	0.727	0.970	0.436
2abk	30%	7×520	F	F	1.000	0.471	0.126

Table 7. SPS of Ref5.

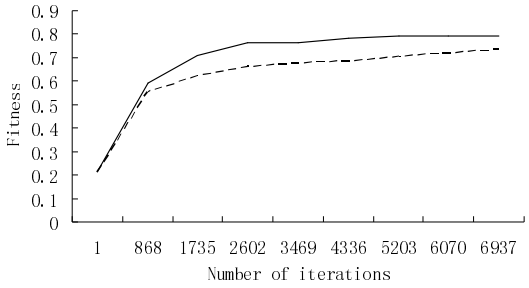
dataset	identity	Seq_no×length	CLUSTAL X	SAGA	DIALIGN	SB_PIMA	QGMALIGN
S51	21%	15×335	0.938	0.831	0.646	0.338	0.363
S52	29%	5×340	1.000	1.000	1.000	0.515	0.789
1eft	19%	8×310	F	F	0.579	F	0.088
1pysA	25%	10×320	0.429	0.429	0.762	0.190	0.176
1qpg	35%	5×510	1.000	0.521	1.000	1.000	0.525
1thm2	38%	7×240	0.774	0.774	1.000	0.194	0.546
Kinase1	26%	5×380	0.806	0.484	0.806	0.677	0.346

CLUSTAL X is a greedy based progressive alignment method. When there are more sequences to be aligned, the major problem with the methods is that mistakes in the initial alignments of the most closely related sequences are propagated to the multiple alignments. The approach doesn't work well on ref4. The DIALIGN program constructs multiple alignment iteratively using the results from segment-to-segment comparisons. It works well on ref4 and ref5, but not very good for ref1 to ref3. SAGA uses twenty-two different genetic operators and each operator has a probability of being chosen - that is to be dynamically optimized during the run. QGMALIGN performs better on ref2 than the other listed methods. In addition, QGMALIGN can compete with CLUSTAL X and SAGA on ref3 and ref4. Experimental results showed that QGMALIGN obtained a better alignment with advantages on global optimization when there are more sequences to be aligned and the length of sequence is nearly 400.

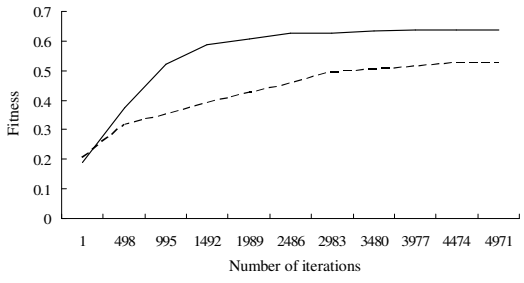
5.2. Comparisons and analysis

To study the effects of the various genetic operators on the alignment, the comparisons of test results of the use of quantum mutation operator and adding genetic operators in it have been made. (See Table 8). The experimental results show that the genetic optimization operators are essential to obtain the better alignment. They can improve the alignment with a lower cost, because the program performs iterations from the 279 per 10 seconds before adding the genetic operators to 282 per 10 seconds after adding the genetic operators on the average.

The quantum rotation angle mutation operator guides the evolutionary process using a single optimal information. Although the optimal solutions of information constantly changes, if the information varies a little, it is easier for the process to fall in the local optimal solution, especially for the difficult multiple sequence alignment problem. The problem is not the unimodal extreme optimization and also the solutions of the problem are not unique. With genetic operators, the QGMALIGN algorithm guides the population towards the optimal solution, while maintaining the diversity of the population during the process of iterations. The added genetic operators improve the efficiency of the algorithm.



(a) the convergent rate for 9rnt



(b) the convergent rate for 1aho

Fig. 9. The convergent rate of the algorithm in two cases.

The dashed line represents the convergent rate of the algorithm with only quantum mutation operator in it.

The real line represents the convergent rate of the algorithm with six genetic operators in it.

The results in Fig. 9 show that the quantum algorithm with six genetic operators performs better on the data 1aho and 9rnt than the pure quantum mutation algorithm.

The algorithm with genetic operators converges faster to the better solution and the quality of the alignment is improved significantly.

6. CONCLUSION

This paper presents the Quantum Genetic algorithm for Multiple sequence ALIGNment - QGMALIGN. The QGMALIGN results show that QGMALIGN performs better on ref2 than the most popular methods (CLUSTALX, SAGA, DIALIGN, SB_PIMA, and QGMALIGN). Also, QGMALIGN can compete with CLUSTAL X and SAGA on ref3 and ref4. If there are a lot of sequences to be aligned and the lengths of sequences are near to 400, then QGMALIGN obtaines the better alignment with advantages on global optimization. The added genetic operators produced a lower cost running time.

Table 8. The experimental results of the algorithm with different operators in it.

sequences	Seq_no×length	All operators		Quantum mutation	
		SPS	#Iterations/time	SPS	#Iterations/time
1plc_ref1	5×100	0.931	4061/28s	0.874	6535/35s
1csy_ref2	19×102	0.684	23479/677s	0.465	27134/789s
1idy_ref3	27×72	0.468	7044/228s	0.320	18633/653s
1pysA_ref4	4×820	0.207	16459/598s	0.103	30000/1223s
1pysA_ref5	10×340	0.176	30000/1375s	0.068	30000/1308s
Average	—	0.493	16209/581.2s	0.366	22460/795.6s

Acknowledgments

This paper was supported by the National Natural Science Foundation of China, Grant No.69601003 and the National Young Natural Science Foundation of China, Grant No.60705004.

References

1. Serafim B. The many faces of sequence alignment. *Briefings in Bioinformatics* 2005; 6(1): 6-22.
2. Thompson JD, Higgins DG, Gibson TJ. CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res* 1994; 22(22): 4673–4680.
3. Barton, G. J. and Sternberg, M. J. E. A strategy for the rapid multiple alignment of protein sequences. *J. Mol. Biol*. 1987; 198: 327–337.
4. Notredame, C., Higgins, D. G. and Heringa, J. T-Coffee: A novel method for fast and accurate multiple sequence alignment. *J. Mol. Biol*. 2000; 302: 205– 217.
5. K., Kuma, K. and Miyata, T. MAFFT: A novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Res*. 2002; 30(14): 3059–3066.
6. Edgar, R. C. MUSCLE: Multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res*. 2004; 32(5): 1792–1797.
7. Van Walle, I., Lasters, I. and Wyns, L. Align-m – a new algorithm for multiple alignment of highly divergent sequences. *Bioinformatics* 2004; 20(9):1428– 1435.
8. Do, C. B., Brudno, M. and Batzoglou, S. ProbCons: Probabilistic consistency-¬based multiple alignment of amino acid sequences. *Genome Research* 2005; 15:330-340.

9. Alexander V. Lukashin, Jacob Engelbrecht, and SØren Brunak. Multiple alignment using simulated annealing: branch point definition in human mRNA splicing. *Nucleic Acids Res*. 1992; 20(10):2511–2516.
10. Alexander V Lukashin, Notredame C, Higgins D G. SAGA: Sequence alignment by genetic algorithm. *Nucleic Acids Res*. 1996; 24(8):1515−1524.
11. K H Han, J -H Kim. Quantum-inspired Evolutionary Algorithm for a Class of Combinatorial Optimization. *IEEE Trans. Evolutionary Computation* 2002; 6(6): 580-593.
12. Hongwei Huo and Vojislav Stojkovic. Applications of Quantum Computing in Bioinformatics. *The 6th annual international conference on computational systems bioinformatics CSB*2007, *Tutorial program PM*2, San Diego, California, August 13-17, 2007.
13. Notredame C, Holm L, Higgins D G. COFFEE: an objective function for multiple sequence alignments. *Bioinformatics* 1998; 14(5): 407-422.