Two-Phase Quantum Based Evolutionary Algorithm for Multiple Sequence Alignment

Hongwei Huo[1] and Vojislav Stojkovic[2]

[1] School of Computer Science and Technology, Xidian University, Xi'an 710071, China
hwhuo@mail.xidian.edu.cn

[2] Computer Science Department, Morgan State University, CA205
1700 East Cold Spring Lane, Baltimore, MD 21251, USA
stojkovi@jewel.morgan.edu

Abstract. The paper presents a two-phase quantum based evolution algorithm for multiple sequence alignment problem,called TPQEAlign. TPQEAlign uses a new probabilistic representation, qubit, that can represent a linear superposition of individuals of solutions. Combined with strategy for the optimization of initial search space, TPQEAilgn is proposed as follows. It consists of two phases. In the first phase, a promising initial value is searched and stored. Each local group has a different value of qubit from other local groups to explore a different search space each. In the second phase, we initialize the population using the stored resulting obtained in the first phase. The effectiveness and performance of TPQEAlign are demonstrated by testing cases in BAliBASE. Comparisons were made with the experimental results of QEAlign and several popular programs, such as CLUSTALX and SAGA. The experiments show that TPQEAlign is efficient and competent with CLUSTALX and SAGA.

1 Introduction

Multiple Sequence Alignment (MSA) is one of the challenging tasks in bioinformatics. It is computationally difficult and has diverse applications in sequence assembly, sequence annotation, structural and functional predictions for genes and proteins, phylogeny and evolutionary analysis. Multiple sequence alignment algorithms may be classified into three classes [1].

The first class is those algorithms that use high quality heuristics very close to optimality [2]. They can only handle a small number of sequences and limited to the sum-of-pairs objective function.

The second class is those algorithms that use the progressive alignment strategy. A multiple alignment is gradually built up by aligning the closest pair of sequences first and then aligning the next closest pair of sequences, or one sequence with a set of aligned sequences or two sets of aligned sequences. This procedure is repeated until all given sequences are aligned together. The best-known system based on progressive multiple alignment is perhaps CLUSTALW. Other multiple alignment systems that are mostly targeting proteins or short DNA sequences, and based on progressive alignment, include MULTALIGN [3], T-COFFEE [4], MAFFT [5], MUSCLE [6], Align-m60 [7], and PROBCONS [8].

The third class of alignment algorithms using iterative refinement strategy can avoid the above problem by aligning these sequences simultaneously. The basic idea is to adopt the evolution theory in nature, initializing a population of individuals of alignments, and then refining these individuals evaluated by an objective function generation by generation, until finding the best alignment. Based on this strategy, SAGA [9], with DIALIGN [10] has become the popular method for multiple alignments.

However, these methods still share some problems, such as local optima, slow convergent speed and lacking a specific termination condition, especially for iterative methods. Some are not flexible enough to capture the full complexity of the similarities between biological sequences.

Quantum evolution algorithm (QEA) is one of the fields of research of Quantum computing. It combines the probabilistic algorithm and quantum algorithm. Kuk-Hym Han has analyzed the characteristics of QEA and showed that QEA can successfully solve the knapsack problem [11]. We try to go one step further and to redesign QEA to solve MSA. We import a variation operator from Genetic Algorithm in QEA, since the representation of the MSA is much more complicated than the knapsack problem.

The paper presents a new Two-Phase Quantum based Evolution Algorithm for multiple sequence alignment, called TPQEAlign - a result of our research on re-designing QEA to solve MSA. The effectiveness and performance of TPQEAlign are demonstrated by testing cases in BAliBASE [12].

2 Multiple Sequence Alignment

Given a finite alphabet set and a set $S = (S_1, S_2, ..., S_n)$ of n sequences with length $l_1, l_2, ..., l_n$, respectively: $S_i = S_{i1}S_{i2}... S_{il}$,$1 \leq i \leq n$, $S_{ij} \in \sum$,$1 \leq j \leq l_i$) where consists of four characters for DNA sequences, and twenty characters of amino acids for protein sequences, a multiple alignment of S is specified by a $n \times l$ matrix $M = (a_{ij})$, $1 \leq i \leq n$, $1 \leq j \leq l$, $l \geq \max(l_i)$, satisfying:

i) $a_{ij} \in \sum \cup$ {-}, where "-" denotes the gap letter;
ii) each row $a_i = a_{i1}a_{i2}...a_{il}$, $1 \leq i \leq n$, of M is exactly the corresponding sequence S_i, if we remove all gap letters;
iii) no column in M contains only gaps.

We can estimate the quality of an alignment by scoring the alignment. The goal of the multiple sequence alignment is to find the optimal alignment that maximizes the score.

3 Algorithms

3.1 Representation

The quantum-inspired evolutionary algorithm deals more efficiently with the balance between exploration and exploitation than traditional genetic algorithm. It explores the search space with a smaller number of individual and a global solution within a shorter span of time.

In quantum computing, the smallest unit of information stored in a two-state quantum.

$$\begin{bmatrix} u \\ v \end{bmatrix}$$

where u and v express the probability amplitudes of the "0" state and the "1" state, respectively. The linear combination of the two basic vectors $|0>$ and $|1>$ can be represented as $u|0> + v|1>$ satisfying the following equation:

$$|u|^2 + |v|^2 = 1 \tag{1}$$

where the probability that the state is measured as basis vector $|0>$ is the square of the norm of the amplitude and the probability that the state is measured as basis vector $|1>$ is the square of the norm of the amplitude, denoted by $|u|^2$ and $|v|^2$, respectively.

A qubit may be in the 1 state, in the 0 state, or in a linear superposition of both states. If there is, for instance, a four-qubits system with four pairs of amplitudes such as

$$M = \left[\begin{array}{c|c|c|c} u_1 & u_2 & u_3 & u_4 \\ v_1 & v_2 & v_3 & v_4 \end{array}\right] = \left[\begin{array}{c|c|c|c} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{2}} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & \frac{\sqrt{2}}{\sqrt{3}} & -\frac{1}{\sqrt{2}} & \frac{\sqrt{3}}{2} \end{array}\right] \tag{2}$$

then the state of the 4-qubits system can be represented as

$$\frac{1}{4\sqrt{3}}|0000> + \frac{1}{4}|0001> - \frac{1}{4\sqrt{3}}|0010> + \frac{1}{2\sqrt{6}}|0100> +$$
$$\frac{1}{4\sqrt{3}}|1000> + \frac{1}{2\sqrt{6}}|1100> - \frac{1}{4\sqrt{3}}|1010> + \frac{1}{4}|1001> -$$
$$\frac{1}{2\sqrt{6}}|0110> + \frac{1}{2\sqrt{2}}|0101> - \frac{1}{4}|0011> - \frac{1}{2\sqrt{2}}|0111> -$$
$$\frac{1}{4}|1011> - \frac{1}{2\sqrt{6}}|1110> + \frac{1}{2\sqrt{2}}|1101> - \frac{1}{2\sqrt{2}}|1111>$$

The probabilities to reach 16 states $|0000>$, $|0001>$, $|0010>$, $|0100>$, $|1000>$, $|1100>$, $|1010>$, $|1001>$, $|0110>$, $|0101>$, $|0011>$, $|0111>$, $|1011>$, $|1110>$, $|1101>$, $|1111>$, are $\frac{1}{48}$, $\frac{1}{16}$, $\frac{1}{48}$, $\frac{1}{24}$, $\frac{1}{48}$, $\frac{1}{24}$, $\frac{1}{48}$, $\frac{1}{16}$, $\frac{1}{24}$, $\frac{1}{8}$, $\frac{1}{16}$, $\frac{1}{8}$, $\frac{1}{16}$, $\frac{1}{24}$, $\frac{1}{8}$, and $\frac{1}{8}$, respectively. Thus, there are possible 2^n states in a system, in which the states are described by n bits. The system M performs a superposition of the four states on each bit independently in sequence and changes the state of the system. Thus, a 4-qubits system comprises the information of 16 states.

For multiple sequence alignment problem, if an alignment of k sequences with the length of N is represented using binary string, it needs a space of $k * N$ binary bits. $k * N$ qubits are used to represent the alignment, which is called qubit alignment individual, denoted by Align-qubit for short.

If, for instance, three sequences *abcd*, *ac*, *abd* are to be aligned,Align-qubit is as follows, where $k = 3$ and $N = 5$ which is the ceiling of 1.2*4, and 4 is the maximum length of the initial sequences. It contains the information of 2^{15} binary states.

$$\left[\begin{array}{c|c|c|c|c} u_{11} & u_{12} & u_{13} & u_{14} & u_{15} \\ v_{11} & v_{12} & v_{13} & v_{14} & v_{15} \\ u_{21} & u_{22} & u_{23} & u_{24} & u_{25} \\ v_{21} & v_{22} & v_{23} & v_{24} & v_{25} \\ u_{31} & u_{32} & u_{33} & u_{34} & u_{35} \\ v_{31} & v_{32} & v_{33} & v_{34} & v_{35} \end{array}\right]$$

The following binary state represents an alignment as:

$$\begin{bmatrix} 0\,0\,0\,0\,1 \\ 0\,1\,0\,1\,1 \\ 0\,0\,1\,0\,1 \end{bmatrix} \longrightarrow \begin{array}{ccccc} a & b & c & d & - \\ a & - & c & - & - \\ a & b & - & d & - \end{array}$$

Binary states that represent a valid binary coding for any alignment are called binary individuals. An Align-qubit individual contains the information of many binary individuals.

3.2 Multiple Sequence Alignment by Quantum Evolutionary Algorithm

QEAlign involves a population consisted of Align-qubit individuals, which can be driven by Q-gate and can collapse to be binary individuals decoded to alignments. Initially, A population of Align-qubit individuals Q(0) is initialized randomly and gives the initial binary individuals P(0) and B(0). In the evolutionary process, the old Align-qubit individuals Q(t-1) is driven by Q-gate to generate the new Align-qubit individuals Q(t), from which generating the new binary individuals P(t) which are optimized by an mutation operator. The binary individuals among P(t) and B(t-1) are evaluated for the fitness value and the best binary individuals among them is stored to B(t). The binary individuals in B(t) is migrated locally or globally under local migration condition or global migration condition, respectively. Then the best binary individual evaluated among B(t) is saved to b. These steps are repeated iteratively, generation by generation. In each generation, good binary individuals survive and bad binary individuals are discarded. The fitness value of b is increased until no more improvement can be made.

All these steps can be grouped as the procedure QEAlign:

```
Procedure QEAlign
1  $t \leftarrow 0$
2  initialize Q($t$)
3  construct P($t$) by collapsing the states of Q($t$)
4  repair P($t$)
5  evaluate P($t$)
6  store the best solutions among P($t$) into B($t$)
7  while (not termination-condition) do
8      $t \leftarrow t + 1$
9      update Q($t$)using Q-gates
10     construct P($t$) by collapsing the states of Q($t$)
11     repair P($t$)
12     mutation P($t$)
13     evaluate P($t$) and B($t$-1)
14     store the best solutions among B($t$-1)and P($t$) into B($t$)
15     store the best solution $b$ among B($t$)
16     if (migration-condition)
17     then migrate $b$ or $b_j^t$ to B($t$) locally endif
18 endwhile
```

The termination condition is that b is not improved after b_{max} times of loops or the number of loops is larger than the given number.

The following in this part is the introduction to the main operations in QEAlign.

Collapsing the states of Q(t) is to construct binary states. In this step, each binary bit of a binary state is set according to the corresponding qubit of Align-qubit individual. For every bit of each binary state, a random number between 0 and 1 is generated, and if the random number is satisfied that $random(0,1) < |\beta_{ij}|^2$, then the bit of this binary state is set to 1, otherwise 0. This process is implemented by the procedure CONSTRUCT(x), where x is a binary state.

```
Procedure CONSTRUCT(x)
1  $i \leftarrow 0$
2  while ($i < nseqs$) do
3     $j \leftarrow 0$
4     while ($j < aln_length$) do
5        if $random(0,1) < |\beta_{ij}|^2$ then $x_{ij} \leftarrow 1$
6           else $x_{ij} \leftarrow 0$ endif
7        $j \leftarrow j + 1$
8     endwhile
9     $i \leftarrow i + 1$
10 endwhile
```

Repair operation is to transform the binary states into be binary individuals such that the number of gaps inserted into any one of the sequences is just equal to $N - n_i$.

Update operation is to update Align-qubit individuals in Q(t) by Q-gate. A Q-gate is acted as a variation operator in QEAlign, the updated Align-qubit should satisfy the normalization condition, $|u'|^2 + |v'|^2 = 1$, by the Q-gate operation, where u' and v' are the values of updated Align-qubit.

In the QEAlign, the following rotation gate is used as Q-gate:

$$U(\Delta\theta_{ij}) = \begin{bmatrix} cos(\Delta\theta_{ij}) & -sin(\Delta\theta_{ij}) \\ sin(\Delta\theta_{ij}) & cos(\Delta\theta_{ij}) \end{bmatrix} \tag{3}$$

```
Procedure REPAIR(x)
1  $i \leftarrow 0$
2  while ($i < nseqs$) do
3     $gapcount \leftarrow aln\_seqlen$
4     while ($gapnum < gapcount$) do
5        $k \leftarrow randint(0, aln\_length)$
6        if ($x_{ik} = 0$) then $x_{ik} \leftarrow 1$ endif
7     endwhile
8     while ($gapnum > gapcount$) do
9        $k \leftarrow randint(0, aln\_length)$
10       if ($x_{ik} = 1$) then $x_{ik} \leftarrow 0$endif
11    endwhile
12    $i \leftarrow i + 1$
13 endwhile
```

and the lookup table of $\Delta\theta_{ij}$ is given in Table1.

Table 1. Lookup table of $\Delta\theta_{ij}$

x_{ij}	b_{ij}	$f_{Cscore}(x_j) \geq$	$\Delta\theta_{ij}$
0	0	false	θ_1
0	0	true	θ_2
0	1	false	θ_3
0	1	true	θ_4
1	0	false	θ_5
1	0	true	θ_6
1	1	false	θ_7
1	1	true	θ_8

where $\Delta\theta_{ij}$ is the function of x_{ij}, b_{ij}, and the expression $f(x_j) \geq f(b_j)$, and x_{ij} is the j-th bit of the i-th sequence of the binary solution x_k^t in P(t), b_{ij} is the j-th bit of the i-th sequence of the binary solution b_k^t in B(t), and b_{ij} is the rotation angle of the the j-th qubit of the i-th row of the qubit individual q_k^t in Q(t). $f_{Cscore}(x_j)$ is the j-th $Cscore$ of the alignment represented by x_k^t and $f_{Cscore}(b_j)$ is the j-th $Cscore$ of the alignment represented by b_k^t. f_{Cscore} is computed as follows.

$$f_{Cscore}(x_j) = C_{score}(s'_{1,i}, s'_{2,i}, ..., s'_{k,i}) = \sum_{1 \le p \le q \le k} P_{score}(s'_{p,i}, s'_{q,i}) \tag{4}$$

where $s'_{1,i}, s'_{2,i}, ..., s'_{k,i}$ is the column of the alignment decoded from x.The process of updating is implemented by the procedure UPDATE:

Procedure UPDATE Q(q)

```
1  $i \leftarrow 0$
2  while ($i < nseqs$) do
3     $j \leftarrow 0$
4     while ($j < aln_length$) do
5         determine $\Delta\theta_{ij}$ according to table 1
6         $[\alpha'_{ij}, \beta'_{ij}] \leftarrow U(\Delta\theta_{ij})[\alpha_{ij}, \beta_{ij}]^T$
7         $j \leftarrow j + 1$
8      endwhile
9      $i \leftarrow i + 1$
10 endwhile
```

QEAlign imports an optional operator (mutation). This operator acts as optimizing the binary individuals. When optimizing a binary individual, we first decode it to be an alignment, then randomly select a block of subsequences, from which generating the template sequence by consisting of the characters with the highest frequency of each column of the subsequences. Template sequence is aligned with each of subsequences by banded-dynamic programming, in which the gaps in each subsequence must be deleted in advance, and template sequences are not inserted gaps when aligning. It is described in the procedure MUTATION(x), where x is a binary individual.

Procedure MUTATION(x)

```
1  Decode $x$ to a alignment
2  Select sub-sequences
3  Find template sequence
4  $i \leftarrow 0$
5  while ($i < nseqs$) do
6      align template sequence and sub-sequence by banded-DP
7      insert sub-sequence in alignment
8      $i \leftarrow i + 1$
9  endwhile
```

A migration in QEAlign is a process of copying b_k^t in B(t) or b to B(t). A global migration is implemented by replaced all the solution in B(t) by b, and a local migration is implemented by replaced some of the solutions in B(t) by the best one of them.

The process of migration is described as the procedure MIGRATION.

```
Procedure MIGRATION(B(t))
1  divided B($t$) into several groups
2  if (global migration condition)
3    then copy $b$ to B($t$)
4      else if(local migration condition)
5              then for each group in B($t$) do
6                      find the best $b_k^t$ in B($t$)
7                      copy $b_k^t$ to the group
8                  endfor
9          endif
10 endif
```

3.3 Two-Phase QEAlign

It has been verified that changing the initial values of qubits can provide better performance of QEA. Since the initial search space is directly determined by the initial values of qubits, the qubit individuals can converge to the best solution effectively if we can seek the initial values of qubits to show the initial search space with small distance to the best solution. Combined with the strategy, TPQEAilgn is proposed as follows.

```
Procedure TPQEAlign
1  First-phase QEAlign
2  Second-phase QEAlign
```

In the first phase of TPQEAlign, all the initial qubit individuals are divided into multiple groups, the initial values of qubit individuals in the same group are initialized as the same value and in different group the initial values are different. In the g-th local group, the initial values of qubits can be decided by the following formula:

$$\begin{bmatrix} u_g \\ v_g \end{bmatrix} = \begin{bmatrix} \sqrt{\frac{(1-2\delta)}{N_g-1} g + \delta} \\ \sqrt{1 - \frac{(1-2\delta)}{N_g-1} g - \delta} \end{bmatrix} \tag{5}$$

where N_g is the total number of groups, δ, $0 < \delta << 1$, a constant. The first phase of TPQEAlign runs without global migration. At the end of the first phase of TPQEAlign, the initial value of qubit individual with the highest fitness is recorded. In the second phase of TPQEAlign, all the initial value of Q(0) is initialized by the recorded value. Then we got a two-phase QEAlign by using the above QEAlign and the idea of TPQEAlign.

4 Experimental Results

The TPQEAlign algorithm was tested on the BAliBASE, a standard Benchmark. And we use SPS to assess the alignment. The following parameters are used in the QEAlign algorithm: population = 100, local group = 5, θ_i, i =1, ... , 8, is given in Table 2. The global migration condition is 100, and the local migration condition is 1.

Table 2. The value of θ

θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8
-0.4π	-0.6π	0.0π	0.1π	0.5π	-0.5π	0.2π	0.5π

Multiple alignment comparisons among CLUSTALW, SAGA, TPQEAlign, and QEAlign with Ref1 through Ref5 are shown in Table 3∼7, where "F" is used to represent the fail alignment.

Table 3. Multiple alignment comparison among CLUSTALW, SAGA, TPQEAlign, and QEAlign with Ref1

Name	ID	CLUSTALX	SAGA	TPQEAlign	QEAlign
1idv	14%	0.705	0.342	0.344	0.194
1havA	15%	0.446	0.411	0.160	0.150
1dox	46%	0.919	0.879	0.835	0.821
1fmb	49%	0.981	0.979	0.948	0.823
2fxb	51%	0.945	0.951	0.956	0.878
9rnt	57%	0.974	0.965	0.915	0.885
1led	43%	0.946	0.923	0.741	0.702
1ppn	46%	0.989	0.983	0.863	0.847

Table 4. Multiple alignment comparison among CLUSTALW, SAGA, TPQEAlign, and QEAlign with Ref2

Name	ID	CLUSTALX	SAGA	TPQEAlign	QEAlign
1aboA	26%	0.650	0.489	0.461	0.347
1idy	28%	0.515	0.548	0.580	0.535
1csy	29%	0.154	0.154	0.581	0.537
1r69	26%	0.675	0.475	0.594	0.587
1tvxA	34%	0.552	0.448	0.630	0.633
1tgxA	35%	0.727	0.773	0.541	0.529
1ubi	32%	0.482	0.492	0.618	0.609
4enl	48%	0.375	0.739	0.745	0.703

Of all the proposed methods, CLUSTALX and SAGA are the most popular methods. Table4 shows that QEAlign and TPQEAlign are better than most of the presented popular aligning methods from Ref2 to Ref4 and not as good as these methods for Ref1 and Ref5. Compared with SAGA, QEAlign is much simpler. It updates the qubit individuals only by one variation operator, while SAGA has operators as many as 22. Moreover, QEA does not need a lot of individuals to search the global optional solution, owing to its qubit representation.

Table 5. Multiple alignment comparison among CLUSTALW, SAGA, TPQEAlign, and QEAlign with Ref3

Name	ID	CLUSTALX	SAGA	TPQEAlign	QEAlign
1idv	20%	0.273	0.364	0.568	0.447
1r69	19%	0.524	0.534	0.416	0.363
1ubi	20%	0.146	0.585	0.351	0.252
1wit	22%	0.565	0.484	0.480	0.432
1ped	32%	0.627	0.646	0.585	0.482
2mvr	24%	0.538	0.494	0.225	0.219
4enl	41%	0.547	0.672	0.569	0.562

Table 6. Multiple alignment comparison among CLUSTALW, SAGA, TPQEAlign, and QEAlign with Ref4

Name	ID	CLUSTALX	SAGA	TPQEAlign	QEAlign
1pvsA	29%	F	0.250	0.352	0.273
1ckaA	19%	F	0.375	0.452	0.349
11kl	28%	1.000	F	0.429	0.354
1vcc	36%	0.485	0.485	0.584	0.524
2abk	30%	F	F	0.490	0.470
kinasel	28%	F	F	0.377	0.340

Table 7. Multiple alignment comparison among CLUSTALW, SAGA, TPQEAlign, and QEAlign with Ref5

Name	ID	CLUSTALX	SAGA	TPQEAlign	QEAlign
1pvsA	25%	0.429	0.429	0.270	0.301
1qpg	35%	1.000	0.521	0.605	0.594
1thm1	32%	0.412	0.765	0.483	0.413
1thm2	38%	0.774	0.774	0.554	0.539
S51	21%	0.938	0.831	0.363	0.353
S52	29%	1.000	1.000	0.573	0.542
kinasel	26%	0.806	0.484	0.520	0.503

5 Conclusions and Future Work

The above analysis follows that QEAlign and TPQEAlign are valid aligning methods. However, QEAlign is not a perfect algorithm for MSA. It does not perform for many test cases. In the future, some better Quantum-gates should be explored for MSA; a new termination criterion is adopted instead of the number of loops; COFFEE is employed as the objective function instead of SPS.

The quantum based techniques described above not only enrich our knowledge of how new computation model can be used for implementing evolutionary algorithm but demonstrate the feasibility of such methods and the novelty of the paradigm.

References

1. Serafim, B.: The many faces of sequence alignment. Briefings in Bioinformatics 6, 6–22 (2005)
2. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. J. Mol. Biol. 147, 195–197 (1981)
3. Barton, G.J., Sternberg, J.E.: A strategy for the rapid multiple alignment of protein sequences. J. Mol. Biol. 198, 327–337 (1987)
4. Notredame, C., Higgins, D.G., Heringa, J.: T-Coffee: A novel method for fast and accurate multiple sequence alignment. Nucleic Acids Res. 302, 205–217 (2000)
5. Katoh, K., Misasa, K., Kuma, K., Miyata, T.: MAFFT: A novel method for rapid multiple sequence alignment based on fast Fourier transform. Nucleic Acids Res. 30, 3059–3066 (2002)
6. Edgar, R.C.: MUSCLE: Multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Res. 32, 1792–1797 (2004)
7. Van, W.I., Lasters, I., Wyns, L.: Align-m - a new algorithm for multiple alignment of highly divergent sequences. Bioinformatics 20, 1428–1435 (2004)
8. Do, C.B., Brudno, M., Batzoglou, S.J.: ProbCons: Probabilistic consistency-based multiple alignment of amino acid sequences. Genome Research 15, 330–340 (2005)
9. Notredame, C., Desmond, G.H.: SAGA: sequence alignment by genetic algorithm. Bull. Math. Biol. 24, 1515–1524 (1996)
10. Burkhard, M.: DIALIGN: multiple DNA and protein sequence alignment at BibiServ. Nucleic Acids Res. 32, W33–W36 (2004)
11. KuK-Hyum, Jong-Hwan, K.: Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. IEEE Transactions on Evolutionary Computation 6, 580–593 (2002)
12. Karplus, K., Hu, B.: Evaluation of protein multiple alignments by SAM-T99 using the Balibase multiple alignment test set. Bioinformatics 17, 713–720 (2001)