Far East Journal of Experimental and Theoretical Artificial Intelligence
Volume 1, Issue 2, 2008, Pages 137-150

INCORPORATING RANDOM PAIRWISE DYNAMIC PROGRAMMING WITH PARTICLE SWARM OPTIMIZATION IN SOLVING MULTIPLE SEQUENCE ALIGNMENT

WANG-SHENG JUANG and SHUN-FENG SU

Department of Electrical Engineering
National Taiwan University of Science and Technology
No. 43, Sec. 4, Keelung Rd. Taipei, 106, Taiwan

Abstract

While solving Multiple Sequence Alignment (MSA) problems, Dynamic Programming (DP) is a commonly-used approach. It is simple and effective. However, when the number of sequences is large, multiple dimensional DP often suffers from large storage and computational complexities. Traditionally, progressive pairwise DP is employed for MSA. It can be expected that such an approach also suffers from local optimal problems. In our previous work, a hybrid algorithm by combining the pairwise DP with the particle swarm optimization (PSO) techniques to overcome the above drawbacks is proposed. The experimental results show promising performance of that algorithm. In this paper, we further propose to consider a random sequence order in aligning pairwise DP progressively. Again, the PSO is employed to avoid the result of alignment being trapped into local optima. From our experiments, it can be found that the proposed algorithm indeed has excellent performance.

Keywords and phrases: multiple sequence alignment, dynamic programming, particle swarm optimization.

Communicated by K. K. Azad

Received June 18, 2008

I. Introduction

In molecular biology, biological sequences are sometimes checked by aligning sequences with each other vertically to show possible similarities or differences among these sequences. The similarities (or commonalties) may reveal evolutionary history and are clues about common biological functions of the sequences. This process is often referred to as the Multiple Sequence Alignment (MSA). MSA may also be employed to construct evolutionary trees from DNA sequences and for analyzing the structures to help in designing new proteins. Generally speaking, MSA is to find an alignment of multiple sequences with the highest score based on a given scoring criterion among sequences. It can be expected that multiple sequence alignment is a combinatorial problem with exponential time complexity and there is no good approach that can solve it efficiently [9].

While solving Multiple Sequence Alignment (MSA) problems, Dynamic Programming (DP) is a commonly-used approach [15]. It is simple and effective. DP converts the original problem to a problem of searching for the shortest path in a weighted directed acyclic k-dimensional graph. Unfortunately, such an approach is notorious for its large consumption of processing time because DP methods with the sum-of-pairs score have been proven to be an NP-complete problem [21]. As a consequence, most of practical multiple sequence alignment algorithms are based on heuristics and usually produce quasi-optimal alignment. Several MSA algorithms have also been reported in the literature [4, 13, 17, 18, 26, 29]. A great majority is to consider the “progressive approach” proposed in [6] or its variation [25]. This approach has the great advantages of speed and simplicity. On the other hand, the main disadvantage is the “local alignment” problem, which stems from the greedy nature of the algorithm. Another kind of approach is to use an extension of DP for simultaneously aligning multiple sequences, such as the Carrillo-Lipman algorithm [2], MSA [11], DCA [22, 23]. In general, these algorithms often have higher quality solutions than those of progressive approaches. However, they have drawbacks of complexity in running time and in memory requirements. Thus, they can only be applied to problems with a limited number of sequences (probably fewer than 10). Another class of approaches used for solving MSA is iterative and stochastic kind of approaches. These approaches include simulated annealing (SA) [14], genetic algorithms (GA) [7, 16, 19, 30] and evolutionary programming (EP) [3, 12, 27]. However, the GA and EP methods introduced so far still suffer from long running time and may not have good search performance. It is because they all start from a random initialization of candidate alignments and therefore spend a lot of time to gradually improve the solutions before reaching a solution near optimal.

In our previous work [8], a hybrid search algorithm referred to as MDPPSO, which combines random pairwise DP and particle swarm optimization (PSO) is proposed for finding solutions for MSA. In that approach, PSO is an improver for a progressive pairwise DP. That approach basically is a pairwise DP based approach and we propose to employ PSO to resolve the local optimum problem. The MDPPSO is an efficient method, but in the initial phase, it needs to calculate all possible pairs' scores. It will generate much computational burden. Thus, in this paper, we propose a new approach referred to as Random Progressive Pairwise Dynamic Programming with Particle Swarm Optimization (RPPDPPSO). In our study, several data sets of Clusters of Orthologous Groups (COGs) [24] of proteins are used as examples to demonstrate that our approach is superior to the most widely used multiple sequence alignment approach ClustalW.

II. Dynamic Programming

In this section, the idea of pairwise dynamic programming is briefly introduced. In our study, the PSO techniques will be embedded into DP to avoid local optima. The related PSO issues will be introduced in the next section. The first use of the Dynamic Programming approach for the alignment of biological sequences was reported in [15]. For a number of useful alignment-scoring schemes, this method is guaranteed to produce an alignment of two given sequences with the highest possible score. There are four steps in a complete DP algorithm, the initialization, step, the Matrix filling step, the Backtracking matrix constructing step and the Alignment obtaining step. The detailed description can be found in [8].

The procedure of DP for finding the maximum score is shown in Algorithm I, where $F(i, j)$ is the maximum score at position (i, j), $\sigma(s_a^i, -)$ denotes a score for a gap in sequence s_a at position i, and $\sigma(-, s_b^j)$ denotes a score for a gap in sequence s_b at position j. The backtracking step is to determine the actual alignment that results in the maximum score. The procedure for constructing the backtracking matrix is described in Algorithm II. The Alignment obtaining step is to obtain the best sequence alignment from the backtracking matrix. A detailed example can be found in [8].

III. Incorporating PSO with DP

Particle swarm theory was first proposed in [5, 10]. Since then, many researchers have employed the theory into the so-called particle swarm optimization (PSO) technique and then apply this technique to widespread areas [1, 20, 28]. PSO is a population based heuristic search technique in which each particle represents a potential solution within the search space and will be characterized by its positions, its velocity and a record of its past individual and global best performance. A modified PSO is

$$v_{id}^{New} = w * v_{id}^{Old} + c_1 * rand(\ ) * (P_{id}^{Old} - x_{id}^{Old}) + c_2 * Rand(\ ) * (P_{gd}^{Old} - x_{id}^{Old}), \quad (1)$$

$$x_{id}^{New} = x_{id}^{Old} + v_{id}^{New}, \quad (2)$$

where w plays the role of balancing the global search and local search, which can be a positive constant or even a positive linear or nonlinear function of time. It is noted that the result of using Eq. (1) to update velocity v_{id} is not an integer value. To cope with this problem, Eq. (1) is modified as follows:

$$\begin{aligned} v_{id}^{New} = round\{&w * v_{id}^{Old} + c_1 * rand(\ ) * (P_{id}^{Old} - x_{id}^{Old}) \\ &+ c_2 * Rand(\ ) * (P_{gd}^{Old} - x_{id}^{Old})\}, \end{aligned}$$

where $round\{\ \}$ is the round-off operation. Particle positions thereby can be updated by Eq. (2). In our implementation of PSO for MSA, each particle in the problem space represents a string of gap positions $X = x_1^1 x_2^1 \cdots x_{n_1}^1 x_1^2 x_2^2 \cdots x_{n_2}^2 \cdots x_1^m x_2^m \cdots x_{n_m}^m$, where x_j^i, for $1 \le j \le n_i$ and $1 \le i \le m$ is the location of a gap existing in sequence i. Here, m is the number of sequences and n_i is the number of gaps for sequence i. n_i is obtained as $n_i = L - l_i$, where l_i is the length of the i-th original sequence and L is the length of sequences used in the algorithm and is determined in the pairwise DP process.

Pairwise DP has a drawback of "once a gap always a gap." If such a gap is improper for the global alignment, it is impossible to modify it in the later DP process. In that case, when more sequences are added into the process, the result obtained will be more far away from the optimal alignment. As mentioned previously, DP for simultaneously aligning multiple sequences has an advantage of resulting in high quality solutions. But, it suffers from large storage and computational complexities, when the number of sequences is large. In fact, we have proposed an approach MDPPSO for solving MSA [8]. It should be noticed that DP is not an initialization mechanism for PSO. In our opinion, a search using an approach of employing DP as an initialization mechanism for PSO may easily be trapped into local optima. In fact, in [12], the author has also used ClustralW as an initialization mechanism for evolutionary programming and the results are not good owing to the local optimum problem. The MDPPSO is an efficient method, but this approach in the initial phase must calculate all possible pairs' scores. If there are n sequences to be aligned, then there are $\frac{n(n-1)}{2}$ possible pairs. To compute all those possible pairs' scores will need lots of computational time. Thus, in this paper, we propose to use randomly selected pairwise DP in the algorithm. The proposed algorithm is referred to as Random Progressive Pairwise Dynamic Programming with Particle Swarm Optimization (RPPDPPSO). The proposed algorithm is shown in Algorithm III.

IV. Experiments and Discussion

In this section, the COG data sets are considered and the obtained results are reported. Table 1 shows all related informations for those data sets. The simulation platform is implemented in MATLAB R11 language, the operating system is Windows XP, with Norton System Works 2003. The processor is Intel Pentium®4 2.5G and the main memory size is 256M. The scoring scheme used is the BLOSUM62 scoring matrix for protein sequences. A pair of gaps with any alphabet gives score –4. A gap to gap pair gives a score of 0. In the process of PSO, the number of particles is 5. The iteration number is 1000. The inertial weight w in the PSO algorithm is set as a random value in the range $[0, 1]$. Parameters C_1 and C_2 are sets 2 and 2, respectively. For the parameters used in PSO, most of them are heuristically selected. In fact, in our study, we simply use a commonly used value and the results are acceptable.

There are many tools being used for MSA. The first category like ClustalW, is to find alignments in a fast way, but the resultant alignments may not be the best solutions. The other category is to employ some optimization search algorithms, such as genetic algorithms, to search for the possibly best alignment. However, this kind of approach may suffer from inefficiency. The proposed approach is compared to those two kinds of approaches. The first one is ClustalW, which is one of the most widely used multiple sequence alignment systems. As mentioned earlier, ClustalW is a progressive approach. The other one is the method proposed in [12]. The method is a stochastic and iterative approach and has been shown to have good search performance. The performance comparisons are shown in Table 2. For smoothing out the randomness of the algorithm, 10 runs of alignment are independently conducted for each data set. The maximum score, the average score and the standard deviation of scores for RPPDPPSO are listed in Table 3. For comparison, the results of using MDPPSO are also included in Table 3. From the results, it can be found that even though MDPPSO has used the order of pairs' scores, the best results are mostly worse than that of using RPPDPPSO. It can be concluded that the order of pair's scores may not be a good choice. By using random orders, although the deviation is large and the average may not be good, the approach provide a chance to find the best solution. The running times of the cases obtaining the best result and the average running time are also listed in Table 4. Notice that "M.C." denotes the numbers of match columns. Finally, the comparison of RPPDPPSO with MDPPSO for running time of the best result and average are shown in Table 5. From those results, it can be clearly seen that the proposed approach in general is better than the other two approaches. The running time is much shorter than those shown in [12] and [8], especially when the number of sequences is large. The ClustalW simulation platform is obtained from the web site http://www.ebi.ac.uk/Tools/clustalw/index.html#, in which alignment result is performed by using default parameters. The experimental results show that RPPDPPSO has better performance.

V. Conclusions

In this paper, an approach of combining modified dynamic programming and particle swarm optimization was proposed for multiple sequence alignment problems. Our previous approach has already proposed this idea. In that approach, when implementing progressive DP, the order of pairs' scores is used. Such an approach may require much computational burden. Besides, such a progressive order may not also be a good choice. Thus, in this paper, we proposed to use a randomly selected order of pairs of sequences. The experimental results reveal that the proposed approach is indeed promising.

References

[1] M. A. Abido, Optimal design of power-system stabilizers using particle swarm optimization, IEEE Transaction on Energy Conversion 17(3) (2002), 406-413.

[2] H. Carrillo and D. J. Lipman, The multiple sequence alignment problem in biology, SIAM J. Appl. Math. 48 (1988), 1073-1082.

[3] K. Chellapilla and G. B. Fogel, Multiple sequence alignment using evolutionary programming, Proceedings of the 1999 IEEE Congress on Evolutionary Computation, 1999, pp. 445-452.

[4] S. C. Chen, A. K. C. Wong and D. K. Y. Chiu, A survey of multiple sequence comparison methods, Bull. Math. Biol. 54 (1992), 563-598.

[5] R. Eberhart and J. Kennedy, A new optimizer using particle swarm theory, Proceedings of the Sixth IEEE International Symposium on Micro Machine and Human Science, 1995, pp. 39-43.

[6] D. F. Feng and R. F. Doolittle, Progressive sequence alignment as a prerequisite to correct phylogenetic trees, Journal of Molecular Evolution 25 (1987), 351-360.

[7] M. Isokawa, M. Wayama and T. Shimizu, Multiple sequence alignment using a genetic algorithm, Genome Informatics 7 (1996), 176-177.

[8] Wang-Sheng Juang and Shun-Feng Su, Multiple sequence alignment using modified dynamic programming and particle swarm optimization, Journal of the Chinese Institute of Engineers 31 (2008), 1-15.

[9] K. Karadimitriou and D. H. Kraft, Genetic algorithms and the multiple sequence alignment problem in biology, Proceedings of the Second Annual Molecular Biology and Biotechnology Conference, 1996, pp. 7.

[10] J. Kennedy and R. Eberhart, Particle swarm optimization, IEEE International Conference on Neural Networks, 1995, pp. 1942-1948.

[11] D. J. Lipman, S. F. Altschul and J. D. Kececioglu, A tool for multiple sequence alignment, Proc. Natl. Acad. Sci. USA 86 (1989), 4412-4415.

[12] K.-H. Liu, Multiple Sequence Alignment: An Evolutionary Programming Based Algorithm with Local Search, Master Thesis, National Taiwan University of Science and Technology, Department of Electrical Engineering, 2003.

[13] M. A. McClure, T. K. Vasi and W. M. Fitch, Comparative analysis of multiple protein sequence alignment methods, Molecular Biology and Evolution 11 (1994), 571-592.

[14] E. W. Myers and W. Miller, Multiple sequence alignment using simulated annealing, Computer Applications in the Biosciences 4(1) (1988), 11-17.

[15] S. B. Needleman and C. D. Wunsch, A general method applicable to the search for similarities in the amino acid sequences of two proteins, Journal of Molecular Biology 48 (1970), 443-453.

[16] C. Notredame and D. G. Higgins, SAGA: sequence alignment by genetic algorithm, Nucleic Acids Research 24(8) (1996), 1515-1524.

[17] C. Notredame, Recent progresses in multiple sequence alignment: a survey, Pharmacogenomics 3(1) (2002), 131-144.

[18] C. Notredame, Recent evolutions of multiple sequence alignment algorithms, PLOS Computational Biology 3(8) (2007), 1405-1408.

[19] M. F. Omar, R. A. Salam, N. A. Rashid and R. Abdullah, Multiple sequence alignment using genetic algorithm and simulated annealing, Proceeding of the 2004 IEEE 12th Conference on Signal Processing and Communications Application, 2004, pp. 28-30.

[20] J. Robinson and Y. Rahmat-Samii, Particle swarm optimization in electromagnetics, IEEE Trans. Antennas and Propagation 52(2) (2004), 397-407.

[21] T. F. Smith and M. S. Waterman, Identification of common molecular subsequences, Journal of Molecular Biology 147(1) (1981), 195-197.

[22] J. Stoye, V. Moulton and A. W. Dress, DCA: an efficient implementation of the divide-and-conquer approach to simultaneous multiple sequence alignment, Computer Applications in the Biosciences 13(6) (1997), 625-626.

[23] J. Stoye, Multiple sequence alignment with the divide-and-conquer method, Gene 211(2) (1998), GC45-GC56.

[24] R. L. Tatusov, E. V. Koonin and D. J. Lipman, A genomic perspective on protein families, Science 278(24) (1997), 631-637.

[25] J. D. Thompson, D. G. Higgins and T. J. Gibson, CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice, Nucleic Acids Research 22(22) (1994), 4673-4680.

[26] J. D. Thompson, F. Plewniak and O. Poch, A comprehensive comparison of multiple sequence alignment programs, Nucleic Acids Research 27(13) (1999), 2682-2690.

[27] R. Thomsen, G. B. Fogel and T. Krink, A Clustal alignment improver using evolutionary algorithms, Proceedings of the 2002 IEEE Congress on Evolutionary Computation 1 (2002), pp. 121-126.

[28] D. W. Van der Merwe and A. P. Engelbrecht, Data clustering using particle swarm optimization, The 2003 IEEE Congress on Evolutionary Computation 1 (2003), pp. 215-220.

[29] I. M. Wallace, G. Blackshields and D. G. Higgins, Multiple sequence alignments, Current Opinion in Structural Biology 15 (2005), 261-266.

[30] C. Zhang and A. K. C. Wong, Toward efficient multiple molecular sequence alignment: a system of genetic algorithm and dynamic programming, IEEE Transactions on Systems, Man, and Cybernetics-part B 27(6) (1997), 918-932.

Table 1. List of datasets

ID	Number of Sequences	Average Length of Sequences (min, max)
COG2178	3	211 (196, 222)
COG1983	4	118 (65, 158)
COG1603	4	222 (199, 245)
COG2157	4	72 (57, 78)
COG1476	5	71 (66, 79)
COG2097	6	96 (81, 113)
COG1510	6	170 (152,185)
COG1761	6	105 (85, 142)
COG0219	9	158 (151, 166)
COG2003	9	206 (148, 243)

Table 2. The comparison with ClustalW, the best result in [12] and the best result in MDPPSO for COG datasets

ID	RPPDPPSO Best Result		ClustalW Result		The Best Result in [12]		The Best Result in MDPPSO	
	Score	M.C.	Score	M.C.	Score	M.C.	Score	M.C.
COG2178	654	44	384	41	653	44	654	44
COG1983	–361	13	–659	11	–323	15	–351	14
COG1603	680	17	149	10	624	16	639	16
COG2157	610	18	499	12	608	18	610	18
COG1476	1677	22	1657	22	1668	21	1674	22
COG2097	2040	12	1781	12	1993	12	1998	12
COG1510	2240	8	1650	4	2157	6	2205	6
COG1761	485	9	–144	7	43	6	423	9
COG0219	10306	26	9734	24	10358	25	10204	25
COG2003	6865	23	5152	18	6442	19	6656	22

Table 3. The comparison with MDPPSO for the maximum, the average and the standard deviation of scores

ID	RPPDPPSO			MDPPSO		
	Max. Score	Average Score	Standard Deviation	Max. Score	Average Score	Standard Deviation
COG2178	654	623.4	31.0	654	647	4.8
COG1983	–361	–379.7	26.3	–351	–361.3	3.9
COG1603	680	636.2	35.8	639	630.1	6.6
COG2157	610	595.2	10.7	610	609.5	0.5
COG1476	1677	1665.5	13.1	1674	1674	0.0
COG2097	2040	1967.5	51.0	1998	1976	15.5
COG1510	2240	2052.4	108.6	2205	2168.1	16.2
COG1761	485	307.6	196.5	423	418.5	2.8
COG0219	10306	10085.0	185.9	10204	10073.7	75.5
COG2003	6865	6562.0	201.2	6656	6586	53.6

Table 4. Running time of the best result and the average running time for COG datasets

ID	Running time of the best result (sec.)	Average running time (sec.)
COG2178	2097	2206.2
COG1983	3386	3528.3
COG1603	5944	6092.0
COG2157	1470	1549.8
COG1476	3019	2969.4
COG2097	7792	7995.4
COG1510	15122	14451.1
COG1761	10171	10013.7
COG0219	38504	40392.7
COG2003	65651	64838.4

Table 5. The comparison with MDPPSO for running time of the best result and average

ID	MDPPSO		RPPDPPSO	
	Running time of the best result (sec.)	Average running time (sec.)	Running time of the best result (sec.)	Average running time (sec.)
COG2178	2097	2206.2	2127	2322.0
COG1983	3386	3528.3	4856	4196.1
COG1603	5944	6092	4605	6551.9
COG2157	1470	1549.8	1279	1455.7
COG1476	3019	2969.4	1922	2076.2
COG2097	7792	7995.4	4471	6105.5
COG1510	15122	14451.1	12453	11493.3
COG1761	10171	10013.7	6263	7531.6
COG0219	38504	40392.7	15153	25860.1
COG2003	65651	64838.4	46459	36200.2

Algorithm I. Dynamic programming for global alignment

Aligning sequences s_a and s_b of length m and n, respectively, with linear gap penalty. Begin

```
Initialization
    for $i := 0$ to $m$ do
        $F(0, i) = -ig$
    end
    for $j := 1$ to $n$ do
        $F(j, 0) = -jg$
    end

matrix fill
    for $i := 1$ to $n$ do
        for $j := 1$ to $m$ do
            $F(i, j) = \max\{F(i-1, j-1) + \sigma(s_a^i, s_b^j), F(i-1, j) + g, F(i, j-1) + g\}$
        end
    end

end
```

Algorithm II. Backtrack matrix construct

Aligning sequences s_a and s_b of length m and n, respectively, with linear gap penalty.

```
begin
for $i := 1$ to $n$ do
   for $j := 1$ to $m$ do
      Up_Value = $F(i-1, j)$
      Left_Value = $F(i, j-1)$
      Up_Left_Value = $F(i-1, j-1)$
      if $(s_a^j := s_b^i)$ do
        BM$(i, j)$ = '*'
      else
         if (Left_Value>= U_Value) do
          if (Left_Value+gap_penalty>=Up_Left_Value+Mismatch) do
             fill BM$(i, j)$ with '–'
           else
             fill BM$(i, j)$ with '*'
           end
         else
           if (Up_Value+gap_penalty>=Up_Left_Value+Mismatch) do
              fill BM$(i, j)$ with '#'
            else
              fill BM$(i, j)$ with '*'
            end
         end
       end
    end
end
end
```

Algorithm III. RPPDPPSO

```
Align multiple sequences $S_1, S_2, ..., S_n$ with BLOSUM62 scoring matrix scheme.
// Combine modified progressive dynamic programming with PSO to align multiple
// sequences in random pair order. Suppose $S_a$ is selected as row sequence and
// $S_b$ is the column sequence.
begin
    To generate a random integer permutation $[p_1\ p_2\ p_3\ ...\ p_n]$
    // $p_{i, i=1, 2, ..., n} \in \{1, 2, ..., n\}$, $p_1 \neq p_2 \neq \cdots \neq p_n$
    Select $S_{p_1}$ as row sequence $S_a$
    for $i := 2$ to $n$ do
        Select $S_{p_i}$ as column sequence $S_b$
        Align $(S_a, S_b)$ using modified DP
        Improving the result of alignment for sequences pair $(S_a, S_b)$ using PSO
        Remove all full spaces column
        Replace $S_a$ with the results of improvement
    end
    Output result of multiple sequence alignment
end
```