Multiple Sequence Alignment by Ant Colony Optimization and Divide-and-Conquer

Yixin Chen[1], Yi Pan[2], Juan Chen[3], Wei Liu[3], and Ling Chen[3]

[1] Department of Computer Science, Washington University in St. Louis, St. Louis, MO 63130, USA
chen@cse.wustl.edu
[2] Department of Computer Science, Georgia State University, Atlanta, GA 30303, USA
pan@cs.gsu.edu
[3] Department of Computer Science, Yangzhou University, Yangzhou 225009, China
lchen@yzcn.net

Abstract. Multiple sequence alignment is a common task in molecular biology and bioinformatics. Obtaining an accurate alignment of protein sequences is a difficult computational problem because many heuristic techniques cannot achieve optimality in a reasonable running time. A novel multiple sequence alignment algorithm based on ant colony optimization and divide-and-conquer technique is proposed. The algorithm divides a set of sequences into several subsections vertically by bisecting the sequences recursively using the ant colony optimization method. We also present two methods that adaptively adjust the parameters and update the pheromones to avoid local optimal traps. Experimental results show that the algorithm can achieve high quality solution and significantly reduce the running time.

1 Introduction

Multiple sequence alignment (MSA) is a common task in bioinformatics. It plays an essential role in detecting regions of significant similarity among a collection of primary sequences of nucleic acid or proteins. MSA is also used to help support the reconstruction of the phylogenetic trees, find the patterns of protein families, detect homology between new and existing sequences, and predict the secondary and tertiary structures of protein sequences.

Given a family $S = (S_1, ..., S_N)$ of *N* sequences, a multiple alignment of *S* is a new set of sequences $S' = (S_1', ..., S_N')$ so that all the strings in *S'* are of equal length and each S_i' is generated from S_i by inserting gaps. To evaluate the quality of the alignment, the *SP* (sum-of-pairs) function is the most popular scoring method. The goal of general multiple sequence alignment algorithms is to find out the alignment with the highest *SP* . MSA based on *SP* scores is an NP problem[1]. For practical reasons the dynamical programming method is only capable of aligning a maximum of a few sequences.

Previous Methods. There are numerous existing methods for MSA. The MSA program [2] can align up to ten closely related sequences. It is an implementation of the

Carrillo and Lipman algorithm [3] that identifies in advance the portion of the hyperspace not contributing to the solution and excludes it from computation. Stoye described a new divide and conquer algorithm DCA [4] sits on top of MSA and expands its capabilities. Recently, OMA, an iterative implementation of DCA [5] is proposed to speed up the DCA strategy and reduce memory requirements. ClustalW [6] is an efficient program based on the progressive algorithm by Feng and Doolittle [7]. Other progressive alignment methods, e.g. Dialign [8,9], assemble the alignment in a sequence-independent manner by combining segment pairs in an order dictated by their score, until every residue of every sequence has been incorporated in the alignment. Prrp proposed by Gotoh [10], AMPS algorithm [11], and Berger and Munsen's algorithm [12] use deterministic iterative methods, while SAGA [13], Gibbs sample algorithm for MSA [14], MSA algorithm based on simulated annealing [15] are stochastic methods. T-Coffee[16], MUSCLE[17], and PROBCONS[18] are integrative algorithms. Other approaches to MSA problem include Hidden Markov Model (HMM), an important method for sequence analysis [19]; POA, a novel MSA algorithm based on the partial order graph[13]; MAFFT, a MSA algorithm based on fast Fourier transform[20]; and the heuristic algorithm based on blocks [21].

Ant Colony Algorithm is a new evolution simulation algorithm proposed by M. Dorigo et al[22]. It successfully resolves the TSP problems by simulating the ants' food-hunting activities. It is recently used to solve the NP-complete problems of job-shop scheduling [23], quadratic assigning[24,25], and sequential ordering [26], and has shown exceptional performance in resolving complex optimization problems especially combinational optimization problem.

In this paper, we present an efficient algorithm for multiple sequence alignment based on ant colony algorithm and a divide-and-conquer strategy. The algorithm divides a set of sequences into several subsections vertically by bisecting the sequences recursively using the ant colony optimization method. The recursive procedure of bisecting the sequences ends when the length of all the sections is equal to one and hence the result of alignment is obtained. In the ant colony algorithm, we also present two methods that adaptively adjust the parameters and update the pheromones so as to avoid local convergence. Experimental results show that the algorithm can get high quality solution and reduce the running time.

2 Aligning Sequence Set by Divide and Conquer

Given a family $S=(S_1,\ldots,S_N)$ of N sequences. Our algorithm first partitions the sequences into several subsets of segments vertically. Each sequence S_i is divided at a suitable character c_i near the midpoint. Here, c_i is a position of a character in S_i or a position between two characters where a gap should be inserted. Therefore we obtain two new families of shorter sequences with one family consist of the prefixes S^p = ($S_1^P(c_1),\ldots,S_N^p(c_N)$) and one of suffixes S^s = ($S_1^s(c_1),\ldots,S_N^s(c_N)$). Consequently, the algorithm partitions the prefix family and the suffix family in a recursive manner to reduce the original multiple sequences alignment problem into more alignment problems involving shorter sequences. The algorithm bisects the sequences using the ant colony optimization method. The recursive procedure of bisecting the sequences ends when the length of all the sections is equal to one and hence the result of alignment is obtained. The algorithm can be recursively described as follows:

```
Algorithm. D&C_Align($S, R$)
Input•$S$=($S_0$,$S_1$,..,$S_{N-1}$): sequences to be partitioned; Output•$R$: alignment of $S$;
begin
  If the maximal length of $S_0$,$S_1$,..,$S_{N-1}$ is larger than 1
    Bisect $S$ into $Sp$ and $Ss$ using the ant algorithm;
    D&C_Align ($Sp$, $Rp$);  D&C_Align($Ss$ , $Rs$);
    $R$= $Rp \cup Rs$;
  end if
end
```

3 Searching for the Cut-Off Points

In the proposed D&C_Align algorithm, we use the ant colony algorithm to iteratively bisect the sequence set S at approximately optimum cut-off points. In the ant colony algorithm, each ant searches for a set of cutting points by starting from the midpoint of a sequence and moving on the other sequences to choose the matching characters. The searching result of the ant population is expressed by a 2-dimensional array d where d_{ij} denotes the position of cut-off of the j-th sequence in the i-th solution.

3.1 Bisecting the Sequence Set

Let $S_0,\ldots,S_{N-1}$ be N sequences. An artificial ant starts from $S_0[m_0]$ which is a character randomly selected in the middle area of S_0, $m_0 \in [mid_0 - \delta, mid_0 + \delta]$, here $mid_0 = \left\lceil \frac{|S_0|}{2} \right\rceil$, δ is the search scope of the ants in S_0. The ant selects one character from or inserts a gap into the middle part of the sequences of $S_1,\ldots,S_{N-1}$ matching with $S_0[0]$. From the sequence S_i, i=1,2,…,n-1, the ant selects a character $S_i[j]$ by a probability determined by the matching score with $S_0[m_0]$, deviation of its location from the middle of S_i, $mid_i = \left\lceil \frac{|S_i|}{2} \right\rceil$, and pheromone on the logical edge between $S_i[j]$ and $S_0[m_0]$. In each step, the ant might select an empty character, which means a gap is inserted into the sequence in the alignment.

The other ants select their path in the same manner, but they start from different sequences. The i-th ant starts from S_i, , and successively goes through S_{i+1}, S_{i+2}, …, S_n. When it reaches S_n, it continues going through S_0, S_1,…,S_{i-1} to complete the path.

At the end of each iteration, the algorithm calculates the fitness of bisecting results of the ants. Then the pheromone on the logical edges is updated according to the fitness of the bisecting result passing through the edge. After a certain number of iterations, the solutions will converge to a near-optimal bisecting point.

3.2 Probability Function for Selecting the Characters

Let $P(k,l,n,m)$ be the probability for the ant starting from $S_k[l]$ to select the character $S_n[m]$ in S_n . We define $P(k,l,n,m)$ as follows:

$$p(k,l,n,m)=\frac{a\times phe(k,l,n,m)+b\times mat(k,l,n,m)+c\times dev(k,l,n,m)}{\sum_{r=mid(n)-\delta}^{mid(n)+\delta}[a\times phe(k,l,n,r)+b\times mat(k,l,n,r)+c\times dev(k,l,n,r)]} \tag{1}$$

Here, $phe(k, l, n, m)$ is the pheromone on the logical edge between $S_k[l]$ and $S_n[m]$, $mat(k, l, n, m)$ is the *SP* score of the characters the ant starting from $S_k[l]$ selected before reaching $S_n[m]$, $loc(k, l, n)$ is the start location in S_n when the ant searches the character matching with $S_k[l]$ within S_n , $dev(k, l, n, m)$ is the location deviation between *m* and the the character which $S_k[l-1]$ has selected in S_n , *a*,*b*,*c* are the weights of pheromone, matching score and location deviation. δ is the range in S_n for the ant selects the character matching with $S_k[l]$.

When the ant selects a character in S_n to match with $S_k(l)$, it first calculates the selecting probabilities for all characters in a certain range of S_n. If $S_n(m)$ has the largest probability and $S_n(m)$ is equal to $S_k(l)$, then the ant selects $S_n(m)$. Otherwise, the ant selects the characters (include the gap) according to their selecting probabilities by the "roulette wheel" method. The characters in S_n will have higher probability to be selected if it has higher pheromone on the logical edge connecting with $S_k[l]$, higher matching score with $S_k[l]$, and less deviation to $mid(n)$.

3.3 Updating the Pheromone

Suppose the *r*-th ant partitions the S_i at position c_i^r into two new families of shorter sequences, one family consisting of the prefixes S^p = ($S_1^P(c_1),\ldots,S_N^p(c_N)$) and one of suffixes S^s = ($S_1^s(c_1),\ldots,S_N^s(c_N)$) are obtained. To evaluate the bisecting result, the fitness of the *r*-th ant is defined as (2) based on our score estimating algorithm SE recently developed [27]:

$$fitness(r)=\sum_{i=1}^{N-1}\sum_{j=1}^{i-1}SE(S_i^p(c_i^r),S_j^p(c_j^r))+\sum_{i=1}^{N-1}\sum_{j=1}^{i-1}SE(S_i^s(c_i^r),S_j^s(c_j^r)) \tag{2}$$

After each ant completes a solution, the pheromone on its logical edges should be updated according the fitness of the solution. Let *evap*1 be the evaporation coefficient, 0≤*evap*1≤1, the value of $phe(k, l, n, m)$ is updated according to formula (3).

$$phe(k,l,n,m)=(1-evap1)\times phe(k,l,n,.m)+evap1\times\sum_{r\in set(k,k,n,m)}fitness(r) \tag{3}$$

Here $set(k,l,n,m)=\{r|\ c_k^r=l,\ c_n^r=m\}$ is the set of ants whose solution includes the logic edge between $S_k[l]$ and $S_n[m]$.

To avoid local convergence, the pheromone on the logical edges should be adjusted adaptively. In the ant algorithm, if the scores of the alignments in iteration are all less than the average score of the alignments in the *d* iterations before, the pheromone on the logical edges should be adjusted. Here d is a parameter which is adjustable according to the number of the iterations:

$$d = k_1 + \left\lceil \frac{k_2}{generation} \right\rceil \tag{4}$$

Here k_1, k_2 are constants, and *generation* is the number of current iteration. It is obvious that in the early iterations d is assigned a greater value since the pheromone is evenly distributed. In the later iterations, the d value becomes smaller since the pheromone might accumulate on a few logical edges. The rule of pheromone adjusting is: if the pheromone is more than a threshold, it should be evaporated according to a coefficient $evap2$ ($0.5 \le evap2 \le 1$) :

$$phe(k,l,n,m) = evap2 \times phe(k,l,n,m) \tag{5}$$

By reducing the pheromone on these logical edges, they have less probability to be chosen by the ants in the later iterations.

3.4 Adaptive Parameters *a*, *b* and *c*

The ants tend to select a path with more pheromone than other paths even if the difference is very slight. As a consequence, the paths the ants select would be concentrated on several paths where the edges have higher pheromone intention. This could result in local optima. To avoid local convergence, we adjust the parameters a, b, and c in probability function (1) adaptively. In the early stages, we set the values of b and c be larger than a so that the heuristic information of matching score and the location deviation have more importance. This helps the ants to search in a wide range to avoid local convergence in the early stages. In later iterations, we should make full use of the feedback information represented by the pheromone. Therefore, we increase a, and decrease b and c adaptively. Combined with the techniques of global and local pheromone updating, the algorithm can successfully speedup the convergence while maintaining the diversity of the solutions to avoid being trapped into local optimal solutions. The initial values of a, b, and c are s_a , s_b, and s_c respectively. At the end of each iteration, the algorithm adjusts a, b, and c as follows:

$$a = a \times (1 + v_a) \qquad if\,(a > t_a) \quad a = s_a \tag{6}$$

$$b = b \times (1 - v_b) \qquad if\,(b < t_b) \quad b = s_b \tag{7}$$

$$c = c \times (1 - v_c) \qquad if\,(c < t_c) \quad c = s_c \tag{8}$$

where v_a , v_b , v_c are rates of update in the range of (0,1), and t_a , t_b , t_c are upper/lower bounds for a, b, and c, respectively.

4 Experimental Result

We test our algorithm D&C_Align using the sequences randomly selected from benchmark database BAliBASE 2.0. As a comparison, we also test SAGA, a multiple sequence alignment program based on an evolutionary algorithm.

Experimental results show that algorithm D&C_Align improves the alignment accuracy for long sequences and requires less computational time than SAGA. The comparison of the running times of D&C_Align and SAGA, on sequences with different lengths is shown in Figure 1.

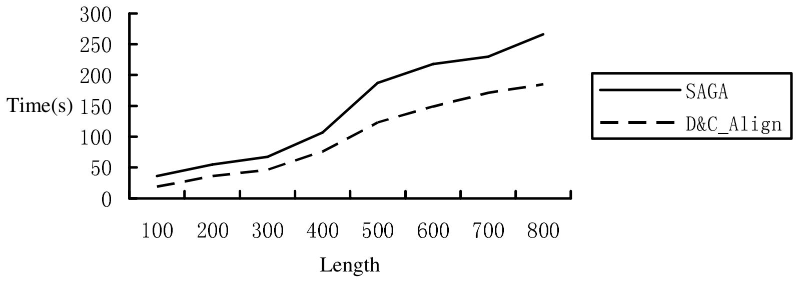

Fig. 1. Comparison of D&C_Align and SAGA on ten sequences

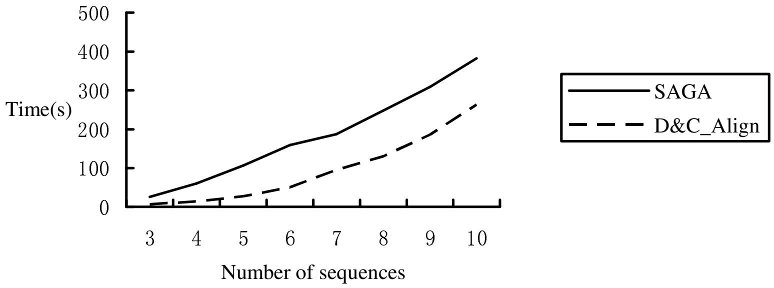

Fig. 2. Comparison of D&C_Align and SAGA of sequences with length of 1000

Figure 2 shows the comparison of the running times of D&C_Align and SAGA on sequences sets with different numbers of sequences, all with 1000 characters.

Comparing with SAGA, the running time of D&C_Align is much faster than that of SAGA. For example, for 4 sequences with 200 characters, the running time of SAGA is 200 to 400 seconds, while the running time of D&C_Align is 10 to 20 seconds.

Table 1 shows the experimental results of D&C_Align on the randomly selected sequences sets from the benchmark database BAliBASE 2.0. The last column in the table is the score of the reference alignment offered by BAliBASE which is very close to the theoretically optimal alignment.

In bioinformatics, if the proportion of similarity in a set of sequences is below 30%, most of the sequence alignment methods cannot find the correct alignment. Therefore, it is called the "twilight zone". The experiment results in Table1 show that D&C_Align can align not only sets of similar sequences but also those sets in the "twilight zone". It can improve the solution quality over the near-optimal reference alignments in a number of cases in Table 1.

Table 1. Experimental results of D&C_Align

Name of sequences	Similarity	Number of sequences	Length of the longest sequence	Running time (s)	Score of D&C_align	Score of reference alignment
1fmb	49	4	106	5	823	828
1r69	12	4	78	3	-274	-289
1tvxA	16	4	69	3	-250	-268
1ubi	16	4	94	4	-298	-342
2trx	15	4	102	4	-301	-293
1idy	13	5	67	3	-282	-285
2fxb	51	5	63	3	760	784
1krn	45	5	82	5	749	814
1uky	17	4	218	11	-740	-731
3grs	17	4	237	12	-778	-768
1bavA	15	5	199	18	-1510	-1508
1bbt3	14	5	192	18	-1403	-1395
1sbp	19	5	263	26	-1199	-1193
1adj	35	4	418	35	301	306
1ajsA	14	4	387	31	-1462	-1458
1lvl	19	4	449	41	-1210	-1214
1pamA	18	5	572	66	-3487	-3443
gal4	14	5	395	45	-2257	-2281
laboA	30	15	80	671	403	790
lidy	19	27	60	3498	-8905	-4701

References

1. Wang L., Jiang T., On the complexity of multiple sequences alignment, Journal of Computational Biology, vol. 1(1994):337-348,
2. Lipman DJ, Altschul SF, Kececioglu JD: A tool for multiple sequence alignment. Proc.Natl. Acad. Sci. USA 86, (1989) 4412-4415.
3. Carrillo H, Lipman DJ: The multiple sequence alignment problem in biology.SIAM J. Appl. Math. Vol. 48, (1988)1073-1082.
4. Stoye J, Moulton V, Dress AW: DCA: an efficient implementation of the divide-andconquer approach to simultaneous multiple sequence alignment. Comput. Appl. Biosci. Vol. 13(6), (1997) 625-626.
5. Reinert K, Stoye J, Will T: An iterative method for faster sum-of-pair multiple sequence alignment. Bioinformatics , vol. 16(9), (2000) 808-814.
6. Thompson, JD, Higgins, DG and Gibson, TJ (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice. Nucleic Acids Research,(1994), vol.22,No.22.4673-4680.
7. Feng D-F, Doolittle RF: Progressive sequence alignment as a prerequisite to correct phylogenetic trees. J. Mol. E, vol. 25, (1987)351-360
8. B. Morgenstern and T. Werner DIALIGN: Finding local similarities by multiple sequence alignment. Bioinformatics vil. 14, (1997)290-294.

9. Morgenstern B DIALIGN 2: improvement of the segment-to-segment approach to multiple sequence alignment. Bioinformatics, , vol.15, no. 3, (1999) 211-218.
10. Gotoh O: Significant Improvement in Accuracy of Multiple Protein Sequence Alignments by Iterative Refinements as Assessed by Reference to Structural Alignments. J. Mol. Biol. Vol.264, no.4, (1996)823-838.
11. Barton GJ, Sternberg MJE: A strategy for the rapid multiple alignment of protein sequences: confidence levels from tertiary structure comparisons. J. Mol. Biol. Vol. 198, (1987)327-337.
12. Berger MP, Munson PJ: A novel randomized iterative strategy for aligning multiple protein sequences. Comput. Appl. Biosci. Vol. 7, (1991)479-484.
13. Notredame C, Higgins DG: SAGA:sequence alignment by genetic algorithm. Nucleic Acids Res. Vol. 24, (1996)1515-1524.
14. Lawrence CE, Altschul SF, Boguski MS, Liu JS, Neuwald AF, Wootton JC: Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. Science vol. 262, (1993)208-214.
15. Kim J, Pramanik S, Chung MJ: Multiple Sequence Alignment using Simulated Annealing. Comp. Applic. Biosci. Vol. 10, no.4, (1994)419-426.
16. Notredame C, Higgins DG, Heringa J T-Coffee: A novel method for fast and accurate multiple sequence alignment. J Mol Biol. 2000 Sep 8;302(1):205-217
17. Edgar RC. MUSCLE: a multiple sequence alignment method with reduced time and space complexity. BMC Bioinformatics. 2004 Aug 19;5(1):113.
18. Do, CB, Brudno, M., and Batzoglou, S. ProbCons: Probabilistic.consistency-based multiple alignment of amino acid sequences. In. Proceedings of the Thirteenth National Conference on Artificial. Intelligence, (2004) 703–708.
19. A.Krogh. An introduction to hidden markov models for biological sequences. In: S. L. Salzberg, D. B. Searls and S. Kasif. Computational Methods in Molecular Biology. Elsevier, (1998) 45-63
20. Kazutaka Katoh , Kazuharu Misawa1 , Kei-ichi Kuma and Takashi Miyata MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. Nucleic Acids Res.vol.30, no.14: (2002)3059-3066
21. A heuristic algorithm for multiple sequence alignment based on blocks. P.Zhao and Tao Jiang J. Combinatorial Optimization vol.5,no.1 (2001)95–115
22. Dorigo M., Maniezzo V., Colorni A. Ant system: Optimization by a colony of cooperating agents [J]. IEEE Transactions on Systems, Man, and Cybernetics-Part B, vol. 26, no.1, (1996) 29-41.
23. Gutjahr J.W. A Graph-based Ant System and its convergence. Future Generation Computer Systems, vol.16, no.8,(2000) 873-888.
24. Talbi E.G.,Roux O.,Fonlupt C.,Robillard D.Parallel Ant Colonies for the quadratic assignment problem. Future Generation Computer Systems, vol.17, no.4 (2001) 441-449.
25. Maniezzo V., Carbonaro A.. An ANTS heuristic for the frequency assignment problem, Future Generation Computer Systems, vol.16, no.8, (2000) 927-935.
26. Gambardella L.M., Dorigo M., Ant-Q: A reinforcement learning approach to the traveling salesman problem, Proceedings of the 11th International Conference on Evolutionary Computation, IEEE Press, (1996) 616-621
27. Yixin Chen, Yi Pan, Ling Chen, Juan Chen, Partitioned optimization algorithms for multiple sequence alignment, To appear in The Second IEEE Workshop on High Performance Computing in Medicine and Biology, Vienna, Austria