Baltic J. Modern Computing, Vol. 2 (2014), No. 2, 75-83

An Adaptive Tabu Search Algorithm for Obtaining Alignment of Multiple Sequences

Chaabane LAMICHE[1], Abdelouahab MOUSSAOUI[2]

[1] Department of Computer Science, Faculty of Mathematics and Informatics, M'sila University, M'sila, Algeria
[2] Department of Computer Science, Faculty of Sciences, Setif University, Setif, Algeria

lamiche07@gmail.com, moussaoui.abdel@gmail.com

Abstract : Multiple sequence alignment (MSA) is one of the most challenging and active ongoing research problems field of computational molecular biology. It is classified as a combinatorial optimization problem, which is solved by using computer algorithms. In this research study, a new method for solving sequence alignment problem is proposed, which is called ATS (Adaptive Tabu Search). This algorithm is based on the classical Tabu Search (TS). ATS is implemented in order to obtain results of multiple sequence alignment. Several ideas concerning neighborhood generation, move selection mechanisms and intensification/diversification strategies for our proposed ATS are investigated. Experiments on a wide range of datasets have shown the effectiveness of the proposed method and its ability to achieve good quality solutions comparing to those given by other existing methods.

Keywords: adaptive tabu search, multiple sequence alignment, neighborhood generation, selection mechanism, tabu search..

1. Introduction

Sequence alignment is one of the most important and challenging problems in computational biology and bioinformatics (Karp, 1993). Finding the optimal alignment of a set of sequences is known as a NP-complete problem (Wang and Jiang, 1994).

Several iterative methods were proposed in the literature to solve MSA problem. The basic idea is to start by an initial alignment and iteratively refines it through a series of suitable refinements called iterations. The process is reiterated until satisfaction of some criteria. Iterative methods can be deterministic or stochastic, depending on the strategy used to improve the alignment. In this way, several metaheuristics have been designed to obtain suboptimal alignments. Metaheuristics have also been applied to solve this problem, for example, Simulated Annealing (Kim *et al.*, 1994), Tabu Search (Riaz *et al.*, 2004), Ant Colony Algorithm (Chen et al., 2007), Genetic Algorithms (Notredame *et al.*, 1997), among others. The disadvantage is that metaheuristics do not guarantee optimal solutions, but solutions generated can be very close to optimal solution in a reasonable processing time.

A brief review of some related works in the multiple sequence alignment field using iterative methods such as tabu search is presented in this section. In (Riaz *et al.*, 2004), authors presented a tabu search algorithm to align multiple sequences. The framework of his work consists to implement the adaptive memory features typical of tabu searches in

order to obtain multiple sequences alignment. Two called aligned and unaligned initial solutions are used as starting points for this version algorithm. Aligned initial solutions are generated using Feng and Doolitles progressive alignment algorithm (Feng and Doolittle, 1987). Unaligned initial solutions are constructed by inserting a fixed number of gaps into different sequences at regular intervals. The quality of an alignment is measured by the COFFEE objective function (Notredame *et al.*, 1998). In order to move from one solution to other, the algorithm moves gaps around within a single sequence and performs block moves. This tabu search uses a recency-based memory structure. Thus, after gaps are moved, the tabu list is updated to avoid cycling and getting trapped in a local solution.

C. Lightner (Lightner, 2008) proposed a several tabu searches which can progressively align sequences. Tabu A is the first tabu search version. An MSA, the basic idea of the Tabu A algorithm was determined by initially aligning the first two sequences using dynamic programming (DP) technique. Then, sequences were added one by one until the entire MSA was obtained. Whenever the tabu search moved to a new solution, the entire MSA was determined using DP and progressively adding on each sequence.

A modified version of Tabu A is Tabu B, that reduced the number of computation required to generate an MSA. For the initial solution, Tabu A divided the MSA up into subgroups. After each subgroup was aligned separately, all the subgroups were aligned together to form an MSA.

One drawback to Tabu A and B was that sequences were progressively added to the alignment using only the pairwise DP procedure. Thus, Tabu A' was implemented to improve the overall SP score of Tabu A. An MSA for Tabu A' was determined by initially aligning the first three sequences using DP. Subgroups of three sequences were each aligned separately. Then, all subgroups with three sequences were progressively aligned together to form an MSA.

Tabu search C is the third proposed version. It used and improved upon the best features from Tabu A, Tabu B and Tabu A', A guide tree, intensification and diversification procedures were new components incorporated into Tabu C. The intensification phase iteratively refines the best MSA, so that gaps introduced early in the alignment can be removed or switched around in the diversification phase, a new MSA is generated that is not in the neighborhood of the current solution.

In this research study, we develop a novel tabu search algorithm called Adaptive Tabu Search (ATS). ATS includes a new technique to generate the neighborhood structure, an efficient intensification phase based descent method to improve solution quality and a restarting search as a diversification strategy to explore other promising regions in the research space of alignments.

The rest of the paper is organized as follows: section 2 presents the classical tabu search algorithm . In section 3, the ATS algorithm is explained in detail. In section 4, the experimentation and results are provided. Finally, the study is conclude in section 5.

2. The Heuristic : Tabu Search

The tabu search is a heuristic that can be used to solve combinatorial optimization problems. It is different from the well known hill climbing local search techniques in the sense that it does not become trapped in local optimal solutions, i.e. the tabu search allows moves out of a current solution that makes the objective function (cost function) worse in the hope that it eventually will achieve a better solution (Al-Sultan, 1995). The tabu search requires the following basic elements to be defined:

- *Configuratio*n: is a solution or an assignment of values to variables.
- *A mov*e: characterizes the process of generating a feasible solution to the combinatorial problem that is related to the current solution (i.e. a move is a procedure by which a new trial solution is generated from the current one).
- *Set of candidate moves:* is the set of all possible moves out of a current configuration. If this set is too large, one could operate with a subset of this set.
- *Tabu restrictions:* these are certain conditions imposed on moves which make some of them forbidden. These forbidden moves are known as tabu. It is done by forming a list of a certain size (*T_size*) that records these forbidden moves. This is called the tabu list. The best tabu list size appears to be problem dependant.
- *Termination Criteria:* one may have noticed that we have not specified in our template a termination criterion. In theory, the search could go on forever, unless the optimal value of the problem at hand is known beforehand. In practice, obviously, the search has to be stopped at some point. The most commonly used stopping criteria in tabu search are

- After a fixed number of iterations (or a fixed amount of CPU time);
- After some number of consecutive iterations without an improvement in the objective function value;
- When the objective function reaches a pre-specified threshold value.

Given the above basic elements, the tabu search scheme can be described as follows: start with a certain (current) configuration, evaluate the criterion function for that configuration. Then, follow a certain set of candidate moves. If the best of these moves is not tabu or if the best is tabu, but satisfies the aspiration criterion, then pick that move and consider it to be the next current configuration; otherwise, pick the best move that is not tabu and consider it to be the new current configuration. Repeat the procedure for a certain number of iterations. On termination, the best solution obtained so far is the solution obtained by the algorithm. The tabu list has a certain size, and when the length of the tabu reaches that size and a new move enters that list, then the first move on the tabu is freed from being tabu and the process continues (i.e. the tabu list is circular) (Al-Sultan, 1995).

In the above paragraph, we have outlined the basic steps of the tabu search procedure for solving combinatorial optimization problems. In the next section, we develop a new algorithm for solving MSA problem based on the above tabu search technique.

3. The Adaptive Tabu Search

3.1. Cost Function

Each multiple sequence alignment algorithm has its own cost function for the alignment of sequences. To be used in sequence alignment, a cost function *C* should be explicitly defined as a measure of overall alignment quality.

In our study a cost function used is COFFEE (Notredame *et al.*, 1998). COFFEE works by first generating the pairwise library of the sequences in the alignment and then it calculates the level of identity between the current multiple alignment and the pairwise library. The global score measuring the quality of the alignment is computed by the following formula.

$$Global\ Objective\ Function = \frac{\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} W_{ij} * Score(A_{ij})}{\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} W_{ij} * Len} \quad . \tag{1}$$

where N is the number of sequences; Len is the length of the multiple alignment; W_{ij} is the percent identity between the two aligned sequences S_i and S_j; A_{ij} is the pairwise projection of sequences S_i and S_j obtained from the multiple alignment; and Score(A_{ij}) is the overall level of identity between A_{ij} and the corresponding pairwise alignment.

3.2. Initial Solution

The generation of an initial solution is an important step towards getting a final improved alignment. A good initial solution can effectively converge faster and hence cut the computational cost. In our developed method, the initial solution is constructed by computing the progressive alignment using an algorithm similar to the one proposed by Feng and Doolittle (Feng and Doolittle, 1987) The three steps to generate the aligned initial solution are as follows:

- Calculate a distance matrix of all N (N-1)/2 pairwise distances for the N input sequences by a global alignment algorithm using Needleman-Wunsch algorithm.
- Construct a guide tree according to the distance matrix by linking the least distant pairs of sequences followed by successively more distant pairs.
- Align each node of the guide tree in the order that it is added to the tree until all sequences have been aligned.

3.3. Neighborhood Generation and Move Selection Mechanisms

Several move mechanisms can be applied to a current alignment to generate a new candidate alignment. Basically, all the move sets are related to change the positions of

the gaps ('-') in the sequences. The proposed move mechanisms to generate neighborhood structure are as follows :

Shuffle (i, j, k, direction): This operation shuffles the left/right (direction) gaps from the gap column (including gap *j*) in the sequence *i* and its left/right (direction) k consecutive characters (Kim and Pramaruk, 1994). The parameters *i, j* and direction in the move sets rules may be randomly determined. But k may be determined by certain distribution function, for example uniformly distribution or inverse function related to the size of k. Only experiment can tell which is the best distribution function for k.

Gap block Move : local changes are facilitated through gap blocks. A gap block is a subsequence consisting of one or more consecutive gaps in one or more aligned sequences. To make this move, a random gap in a random sequence is picked. Then the gap block is extended vertically through the other sequences containing a gap at that position. Afterwards, the gap block is extended horizontally to both sides if all the chosen sequences contain a gap there. Finally, the gap block is moved to a randomly chosen new position in the alignment (Lindgreen1 and Krogh1, 2007). The procedure is illustrated in Figure 1 below :

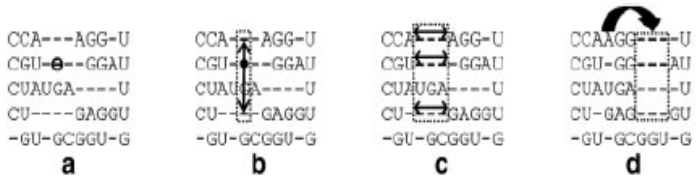

Figure. 1. Illustration of the gap block moves used in our proposed approach. (a) choose a gap at random in a sequence, (b) extend vertically, (c) extend horizontally and (d) move to new position in alignment.

We have to resort such level of move generation that helps to complete the alignment process in reasonable time. In a typical arrangement, in each iteration the algorithm generates one single sequence move for each sequence by Shuffle mechanism and the block moves comprising every possible block of gaps in the alignment. The moves are generated is a stochastic fashion. In case of single sequence moves, the patch of gap(s) as well as the its new location in the sequence are determined randomly. For block moves, every rectangular block of gaps is moved to some random location, either left or right to its current position.

3.4. Tabu list

The size (*or tabu list length: TLL*) of the tabu list is one of the important features of tabu search. It is considered its ability to avoid being trapped in local optima. The size of a tabu list can affect the search performance. Although a longer list may prevent cycling, it requires more scanning and may limit the search domain. To avoid this problem, in our developed adaptive tabu search method we chose changing a tabu list size dynamically during the search as follows: starting by $TLL = [\sqrt{N}]$ value. This size may be modified to $TLL = [\sqrt{N}] + k$ or $TLL = [\sqrt{N}] - k$ during the search process and *k* is fixed experimentally.

For example, if no improvement after *k=10* iterations, it is necessarily to increase or decrease the *TLL* value using the number of sequences *N* and *k*.

We indicate here, that this technique which is used to modify the *TLL* value is considered as another way for applying intensification and diversification strategies respectively.

3.5. Aspiration Criteria

While central to the tabu search method, tabus are sometimes too powerful. They may prohibit attractive moves, even when there is no danger of cycling, or they may lead to an overall stagnation of the search process. It is thus necessary to use algorithmic devices that will allow one to cancel tabus. These are called aspiration criteria. In our approach we choose the simplest and most commonly used aspiration criterion, allows a tabu move when it results in a solution with an objective value better than that of the current best-known solution (since the new solution has obviously not been previously visited).

3.6. Intensification and diversification Strategies

The idea behind the concept of search intensification is that, as an intelligent human being would probably do, one should explore more thoroughly the portions of the search space that seem "promising" in order to make sure that the best solutions in these areas are found. A technique that is used in our work consists to applied a descent algorithm in order to find a local optimum whenever an increase of the objective function is followed immediately by a decrease. The descent algorithm uses with the two neighborhood mechanisms which are cited in section 4.3. Because the tabu list is ignored during the descent procedure, the result is an intensified search to a local optimum. However, to avoid disruption to the tabu search algorithm, the search continues from the sequence that is input into the descent step, rather than from the local optimum sequence that is output.

Diversification is an algorithmic mechanism that tries to improve solution by forcing the search into previously unexplored areas of the search space. It is usually based on some form of long-term memory of the search. In our ATS method we retain the called restart diversification strategy, this one involves introducing a few rarely used components in the current solution (or the best known solution) and restarting the search from this point. In our study we can swapping gaps positions randomly in the best known solution and restarting search from this point.

3.7. Termination Criteria

The last element necessary for tabu search is termination criteria. In our study, search can be stopped when certain number of iterations *ITMAX* is completed. The pseudo–code of our Adaptive Tabu Search (ATS) is given below :

Pseudo-code of ATS algorithm :

Begin

Step 0: Generate an initial aligned alignment A_{INIT}.

Step 1: Set the current alignment $A_{CURR} = A_{INIT}$ and the best alignment $A_{BEST} = A_{INIT}$.

Step 2: Iterate the following steps for *ITMAX* iterations

Step 2.1: Generate N_S neighboring alignments A_i ($i=1,2 \ldots, Ns$) of the current solution A_{CURR} using neighborhood generation mechanisms mentioned above evaluate their corresponding objective function values J_i ($i=1,2 \ldots, Ns$).

Step 2.2: Sort the objective function values off all N_S neighbors in decreasing order. From the best test alignment to the worst test alignment, if the test alignment is non-tabu alignment or it is a tabu alignment but its objective function value is greatest than that of the best alignment A_{BEST} choose this alignment as the current alignment A_{CURR} go to step 2.3; otherwise try next test alignment. If all test solutions are tabu alignments, go to step 2.1.

Step 2.3: If the function objective value of A_{CURR} is greater than that of A_{BEST} set $A_{BEST} = A_{CURR}$

Using intensification phase to improve A_{BEST}

Swapping gaps positions randomly in the A_{BEST} solution to create New_A_{BEST} and restarting search from $A_{CURR} = New\_A_{BEST}$

Step 2.4: If the tabu list is full, remove the oldest alignment in the tabu list.

Step 2.5: Insert the current alignment A_{CURR} into the tabu list.

Step 3: Record the best alignment A_{BEST} and terminate the algorithm.

End.

4. Experimental Results

The algorithm is implemented using JCreator version 3.5 and personnel computer with 2.66 GHz Intel Pentium IV processor. In our first experiment, we have used sequence data sets provided by BAliIBASE. (Thompson *et al.*, 2001). In order to prove the performance of our Adaptive Tabu Search approach we performed a set of tests where the results of ATS are compared with a simple iterative search algorithm and the tabu search described in (Riaz *et al.*, 2004). The simple iterative search algorithm works in a similar fashion as tabu search using COFFEE as objective function and terminates when there is no improvement in quality of the alignment for a specified number of iterations, except that it does not implement the features like tabu list, tabu tenure, aspiration and intensification/diversification. The BAliIBASE scores which are described in (Riaz *et al.*, 2004) are used to evaluate our study and the results of comparison using computed BAliIBASE scores are shown in table 1 and table 2 below:

Table 1. Comparative results for easy to align sequences.

Test case	Tabu Search (Riaz *et al.*, 2004)	Simple Iterative Search (Riaz *et al.*, 2004)	Our Proposed Tabu Search
1csy_ref1	0.805	0.767	**0.930**
1ycc_ref1	0.927	0.879	**0.945**
1pgtA_ref1	0.948	0.590	**0.96**
1ldg_ref1	0.908	0.754	**0.938**
1mrj_ref1	0.994	0.939	**0.996**
1pii_ref1	0.875	0.718	**0.923**
2cba_ref1	0.843	0.715	**0.890**
3grs_ref2	0.842	0.842	**0.967**
1idy_ref3	0.946	0.882	**0.977**
1pysA_ref4	0.500	0.500	**0.735**
kinase2_ref5	0.793	0.787	**0.875**
1pysA_ref5	0.560	0.522	**0.703**
Average	**0.828**	**0.741**	**0.903**

Table 2. Comparative results for hard to align sequences.

Test case	Tabu Search (Riaz *et al.*, 2004)	Simple Iterative Search (Riaz *et al.*, 2004)	Our Proposed Tabu Search
1idy_ref1	0.419	0.208	**0.656**
1tgxA_ref1	0.712	0.700	**0.804**
451c_ref1	0.62	0.543	**0.870**
1ton_ref1	0.719	0.633	**0.952**
2pia_ref1	0.648	0.333	**0.823**
1havA_ref1	0.359	0.118	**0.515**
2hsdA_ref1	0.654	0.273	**0.788**
kinase_ref1	0.749	0.403	**0.958**
1gdoA_ref1	0.823	0.609	**0.965**
Average	**0.634**	**0.424**	**0.814**

The results in table 1 and 2 show clearly the considerable improvement of scores by the proposed tabu search. Indeed, that improved tabu search produces alignments of much better quality than that of simple iterative search in both the specified categories.

5. Conclusion

Multiple sequence alignment is an extension of pairwise alignment to incorporate more than two sequences at a time. In this research, we present an adaptive tabu search method to the MSA problem. Our multiple alignment method try to align all of the sequences in a given query set. The results reported in this paper show the superior capability of our improved tabu search compared to others of the literature. Efficient mechanisms to neighborhood structure generation and other features such as intensification and diversification strategies are the key of this improvement.

As a perspective of this work, the improvement of the start solution by a specific heuristic is desired. In addition we can integrate other mechanisms to neighborhood generation step or incorporate other strategies in intensification/diversification phase to improve the quality of multiple alignment. A comparison of the proposed method with some other stat-of-the-art techniques such as Clustal, SAGA or MULTALIGN is possible to verify its effectiveness.

References

Al-Sultan, K. S. (1995). A Tabu search approach to the clustering problem, *Pattern Recognition* l, 1443–1451.

Chen, L. Zou, L. and Chen, J. (2007). An efficient ant colony algorithm for multiple sequences alignment, *Proc. 3rd International Conference on Natural Computation*, 208-212.

Feng, D. and Doolittle, R. (1987). Progressive Sequence alignment as a prerequisite to correct phylogenetic trees, *Journal of molecular Evolution* 25, 351-360.

Karp, R. M. (1993). Mapping the genome: Some combinatorial problems arising in molecular biology, *Proc. 25th Annual ACM Symposium on the Theory of Computing*, 278-285.

Kim, J. Pramanik, S. and Chung, M. J. (1994). Multiple sequence alignment using simulated annealing, *Computer Applications in the Biosciences* 10, 419-426.

Kim, J. and Pramaruk, S. (1994). An Efficient method for multiple sequence alignment, *Proc. ISMB*, 212–218.

Lightner, C.: (2008). A Tabu search approach to multiple sequence alignment, Ph.D. dissertation, North Corolina State University, Raleigh, North Corolina.

Lindgreen1, S. P. and Krogh1, A. (2007). MASTR: Multiple alignment and structure prediction of non-coding rnas using simulated annealing, *Bioinformatics* 23, 3304-3311.

Needlman, S. B. and Wunsch, C.D. (1970): A General method applicable to the search for similarities in the amino-acid sequence of two proteins, *Journal of Molecular Biology* 48, 443-453.

Notredame, C. (2002). Recent Progresses in MSA: A Survey, *Pharmacogenomic* 3, 1-14.

Notredame, C. and Higgins, D. G. (1996). SAGA: Sequence alignment by genetic algorithm, *Nucleic Acids Research* 24: 1515-1524.

Notredame, C., Holm, L. and Higgins, D. G. (1998). COFFEE: An objective function for multiple sequence alignments, *Bioinformatics* 14, 407–422.

Riaz, T. Wang, T. Y. and Li,. K. B. (2004). Multiple sequence alignment using tabu search, *Proc. 2nd Asia-Pacific Bioinformatics Conference*, Dunedin, New Zealand, 223-232.

Thompson, B. A., Thierry, J. D. and Poch, O. (2001). BAliBASE (Benchmark Alignment dataBASE): Enhancements for repeats, transmembrane sequences and circular permutations. *Nucleic Acids Res*. 29, 323-326.

Wang, L. and Jiang, T. (1994). On the complexity of multiple sequence alignment, *Journal of Computational Biology* 1, 337-348.

Received April 30, 2014, revised June 16, 2014, accepted June 25, 2014