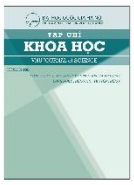
VNU Journal of Science: Comp. Science & Com. Eng, Vol. 34, No. 1 (2018) 1-9

VNU Journal of Science:
Computer Science and Communication Engineering
Journal homepage: http://www.jcsce.vnu.edu.vn/index.php/jcsce



A new Memetic Algorithm for Multiple Graph Alignment

Tran Ngoc Ha[1,3], Le Nhu Hien[2], Hoang Xuan Huan[3,*]

[1]*Thai Nguyen University of Education, 20 Luong Ngoc Quyen, Thai Nguyen, Thai Nguyen, Vietnam*
[2]*Hanoi University of Industry, 298 Cau Dien, Bac Tu Liem, Ha Noi, Vietnam*
[3]*VNU University of Engineering and Technology, 144 Xuan Thuy, Cau Giay, Hanoi, Vietnam*

Abstract

One of the main tasks of structural biology is comparing the structure of proteins. Comparisons of protein structure can determine their functional similarities. Multigraph alignment is a useful tool for identifying functional similarities based on structural analysis. This article proposes a new algorithm for aligning protein binding sites called ACOTS-MGA. This algorithm is based on the memetic scheme. It uses the ant colony optimization (ACO) method to construct a set of solutions, then selects the best solution for implementing Tabu Search to improve the solution quality. Experimental results have shown that ACOTS-MGA outperforms state-of-the-art algorithms while producing alignments of better quality.

Received 08 March 2018, Revised 21 May 2018, Accepted 28 May 2018

Keywords: Multiple Graph Alignment, Tabu Search, Ant Colony Optimization, local search, memetic algorithm, SMMAS pheromone update rule, protein active sites.

1. Introduction

The functional inference of unknown proteins through known proteins plays an important role in the field of life sciences in general and in the field of pharmaceutical chemistry in particular. In this study, comparison of proteins plays a central role.

Prediction of protein function can be executed at both the sequence level and the structural level. Recognizing that proteins with an amino acid sequence similarity more than 40% often have similar functions [1], so comparison at sequence level is the first method used. Many diference approaches are introduced and widely used [2-7]. However, these methods are not suitable for determining inter-molecular functional similarity because functitionality is more closely associated with structures specific than sequential features [6, 12, 16, 18].

To analyze proteins structure, some authors [9, 12-18] proposed using graph model to represent the three-dimensional structure of the protein. Recent studies are based on the Cavbase database [19, 20]. Graph alignment techniques are used to identify functional similarities based on structural analysis. The first methods mainly relie on techniques that exact matching the pairs of graphs. These studies have yielded significant results when studying the functional evolution of non-homologous molecules. However, it is difficult to apply these techniques to discover of

[*] Corresponding author. E-mail: huanhx@vnu.edu.vn.
https://doi.org/10.25073/2588-1086/vnucsce.194

meaningful biological patterns that are approximately conserved.

In order to overcome the disadvances of graph matching methods, the multiple graph alignment problem (MGA) was first proposed by Weskamp et al [21] in 2007. They used it for structural analysis of protein active sites. They also proposed a heuristic algorithm to solve this problem.

MGA was proven to be NP-hard problem [8, 21]. The heuristic algorithms are only suitable for small size problems, so they are not suitable for real applications. Fober et al [8] have extended the usage of MGA problem for the structural analysis of biomolecules and have proposed an evolutionary algorithm called GAVEO. Experiments show that this algorithm is more efficient than greedy algorithm although it is more time consuming.

In [22] we proposed ACO-MGA algorithm that using simply ant colony optimization scheme to solve the multiple graph alignment problem. Experiment shows that this algorithm has better results than the GAVEO algorithm. However, its runtime is long and its efficiency is not good for large data sets.

Memetic algorithm was introduced by Moscato in 1989[23]. It introduces local search techniques for iterative algorithms based on population. The solutions found after each iteration are selected upon to apply the local search techniques in a flexible way. Recently, the algorithms based on this framework are efficient applied in field of bioinformatics [24–26]. In [27] we proposed a two-stage memetic algorithm to solve MGA problem called ACO-MGA2. This algorithm based on ACO algorithm, but it has some changes: the first change is the way to calculate heuristic information, the second one is that local search procedure is applied only in the second stage of algorithm to decrease runtime. Experiments on real datasets have shown that ACO-MGA2 produced better solution quality than ACO-MGA and GAVEO. Because the local search procedure is only executed in the second stage, ACO-MGA2 runs faster than ACO-MGA.

This paper introduces a new two-stage memetic algorithm based on ant colony optimization called ACOTS-MGA (Ant Colony Optimization and Tabu Search for Multiple Graph Alignment) as an improvement of the ACO-MGA2 to solve MGA problem. We keep construction graph as in ACO-MGA2, but improve the random walk procedure, heuristic information and the local search procedures. The local search is replaced by Tabu Search. It only applied at the second stage of the memetic scheme [23]. Improvements in solution quality of ACOTS-MGA is demonstrated empirically by comparison with GAVEO and Greedy.

The rest of this paper is organized as follows: Section 2 provides mathematical statements for multiple graph alignment problem. Section 3 introduces the proposed algorithm. The experimental results are presented in Section 4. Several conclusions are presented in the last section.

2. Problem statement

2.1. Modeling protein binding sites as graphs

The studies [8, 21, 22, 27] are based on the Cavbase database [19]. In this database, the binding pockets are approximately presented by graphs [19, 20]. Each binding pocket is represented by a graph G (V, E), where V is the set of labeled vertices and E is the weighted edges set. A vertex of graph is called as *pseudocenter*. The pseudocenter represented the arrangement in the space and the phisicochemiscal properties of a binding pocket. The labels of the vertites belong to a labeled set L = {A,B,C,D,E,F,G}, where A stands for donor, B for acceptor, ... Two centers are considered the connection and represented by an edge in G if the euclidean distance of them is less than 12 Å. Its label is the weight *w(e)* of it.

In each graph, there are three edit operations:

i) Insertion or deletion of a node: A node $v \in V$ and edges associated with it can be deleted or inserted.

ii) Change of the label of a node: The label $l(v)$ of a node $v \in V$ can be replaced by other label in L.

iii) Change of the weight of an edge. The weight $w(e)$ of an edge e can be changed based on the conformation.

The edit distance of two graphs, G_1 and G_2, is defined as the cost of a cost-minimal sequence of edit operations to transform graph G_1 to G_2. As in sequences alignments, this allows for the introduction of the concept of an alignment of two (or more) graphs. Corresponding to the gaps in sequence alignment, the dummy nodes is defined as placeholders of deleted nodes.

2.2. Multiple graph alignment problem

To study proteins characteristics, Weskamp et al introduced the multiple graph alignment problem [21].

Multigraph is defined as a set of *n* graphs $G = \{G_1(V_1, E_1),\ldots, G_n(V_n, E_n)\}$, where $G_i\ (V_i, E_i)$ is a connected graph, each vertex is labeled under a given set L, and the edges weight represent the Euclidean distances between the vertices.

Call V_i^* is a set of vertices that is created by add a dummy node (denoted ⊥) to set V_i. Dummy node is a node that is not connected to the other nodes. Then $A \subseteq V_1^* \times V_2^* \ldots \times V_n^*$ is an alignment of multigraph G if and only if:

i) For all $i=1,\ldots,n$ and for each $v \in V_i$, there exists exactly one column vector $a^j = (a_1^j,\ldots,a_n^j)^T \in A$ such that $v = a_i^j$

ii) For each column vector $a^j = (a_1^j,\ldots,a_n^j)^T \in A$, there exists at least one $1 \le i \le n$ such that $a_i^j \neq \perp$.

Each $a^j = (a_1^j,\ldots,a_n^j)^T \in A$ ($1 \le j \le m$, m is the number of vertices of the graph with the highest number of vertices) is called a column vector at column j of corresponding alignment matrix A, $v \in V_i$ is a real node.

Figure 1 is an example of MGA. Mutual assignments of nodes are indicated by dashed lines. Note that the third assignment involves a mismatch node, since the label of node in the fourth graph is D. Likewise, the fourth assignment involves a dummy node (indicated by a box), since a node is missing in the third graph.

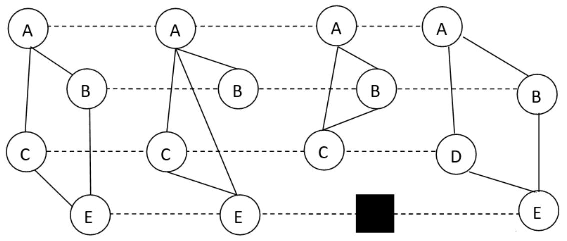


Figure 1. A simple illustration of MGA by an approximate match of four graphs.

For readers' ease, we call $G^* = \{G_1^*(V_1^*, E_1), G_2^*(V_1^*, E_2),\ldots, G_n^*(V_n^*, E_n)\}$ to refer to the multigraph in which the graph G_i has been added a dummy node.

The main task of an MGA problem is to find an alignment $A = (a^1,\ldots, a^m)$ that maximizes the scoring function $s(A)$.

$$s(A) = \sum_{i=1}^{m} nodeScore(a^i) + \sum_{1 \le i < j \le m} edgeScore(a^i, a^j) \quad (1)$$

where *nodeScore* calculated by the equation 2 evaluates the correspondence of all mutually assigned nodes in a column a^i of matrix the alignment. Matching node labels rewarded by a positive value ns_m, mismatches or the alignment of dummy node are penalized by negatives values ns_{mm} and ns_{dummy} respectively.

$$nodeScore\begin{pmatrix} a_1^i \\ \vdots \\ a_n^i \end{pmatrix} = \sum_{1 \le j < k \le n} \begin{cases} ns_m & l(a_j^i) = l(a_k^i) \\ ns_{mm} & l(a_j^i) \neq l(a_k^i) \\ ns_{dummy} & a_j^i = \perp,\ a_k^i \neq \perp \\ ns_{dummy} & a_j^i \neq \perp,\ a_k^i = \perp \end{cases} \quad (2)$$

and *edgeScore* evaluates the compatibility of the edge weights. Tolerance towards edge weights deviation is again realized by ε threshold. Hence, the assignment of two edges is considered a match, if respective weights deviate by ε at most, and otherwise is mismatches. *edgeScore* of two column a^i and a^j

of the alignment matrix A is calculated by the equation 3:

$$edgeScore\left(\begin{pmatrix} a_1^i \\ \vdots \\ a_n^i \end{pmatrix}, \begin{pmatrix} a_1^j \\ \vdots \\ a_n^j \end{pmatrix}\right) = \qquad (3)$$

$$\sum_{1 \le k < l \le n} \begin{cases} es_{mm} & (a_k^i, a_k^j) \in E_k, (a_l^i, a_l^j) \notin E_l \\ es_{mm} & (a_k^i, a_k^j) \notin E_k, (a_l^i, a_l^j) \in E_l \\ es_m & d_{kl}^{ij} \le \varepsilon \\ es_{mm} & d_{kl}^{ij} > \varepsilon \end{cases}$$

In Equation 3, $d_{kl}^{ij} = |w(a_k^i) - w(a_l^j)|$. Parameters ($ns_{m,}$ ns_{mm} ,ns_{dummy} , es_m , es_{mm}) are constants used to reward or penalize matches, mismatches and dummies, respectively. In this article, they are initialized as same as in [8]: ns_m = 1.0; ns_{mm} = -5.0; ns_{dummy} = -2.5; es_m = 0.2; es_{mm} =-0.1.

Call V_{max} is the number of vertices of the graph with the highest number of vertices and n is the number of graphs. Because MGA is a NP-hard problem (see [8, 21]), so its complexity will be $\boldsymbol{O}((\boldsymbol{Vmax})!^{\boldsymbol{n}})$ if we use the exhaustive method to solve it

3. The proposed algorithm

The proposed algorithm based on the ACO algorithm. It combines the ACO with Tabu Search procedure arcording to the memetic scheme. An algorithm based on the ant colonies optimization method has four important components: construction graph, heuristic information, pheronome update rules, and local search procedure. These components of ACOTS-MGA are presented as follows.

3.1. Components of ACOTS-MGA

a) Construction Graph

The construction graph consists of n layers where layer *i* is graph G_i^* in the set G^*. Each vertex of the above layer is connected to all of vertices of the next below layer. The top layer considered as the next layer of the bottom layer.

Figure 2 illustrates the construction graph where ants start from the graph G_1 which does not display edges within a graph, white nodes are real vertices and grey nodes are dummy.

An alignment of graphs is a path from G_1 through every layer to G_n such that each path passes only one vertex of each layer and each vertex of the construction graph has only one path passing through. Dummy nodes allow more than one paths to passes through.

Remark. Note that the paths forming this alignment can be considered as a single path by the insight of the popular ACO algorithm. This implied path starts from a vertex of the graph G_1 passing through all next graphs to the last graph. It then "walks" to the vertex of the top layer of another alignment vector until passing through all real nodes, each node exactly once time.

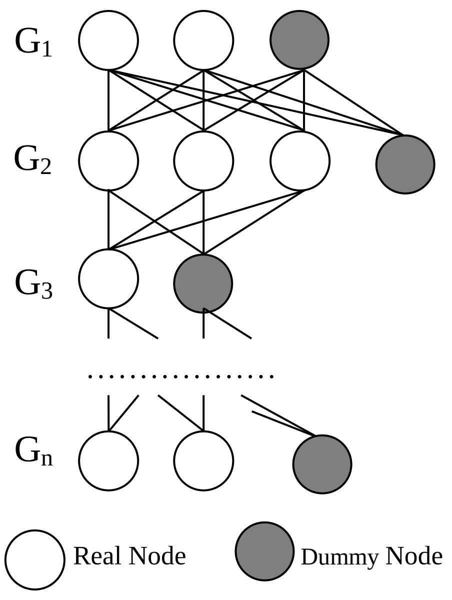


Figure 2. Construction graph for n-graphs alignment.

b) Heuristic information

Heuristic information $\eta_{j,k}^i(a^j)$ is the node score. It is calculated by equation (2) when aligning node k of graph G_i at position i of column vector a^j.

c) Random walk procedure to construct an alignment

In each iteration, each ant will repeat the process to build vectors $a^j = (a_1^j, ..., a_n^j)^T$ for an alignment A as follows.

The ant selects randomly one vertex on the first layer as initial vertex. At the next layers, difference with ACO-MGA2 which consider all vertices of graph G_i to choose a vertex to align, in ACOTS-MGA, the aligned node is chosen by beam search strategy. This stratery helps ACOTS-MGA decrease time to indentify node to align. This procedure is described as follows.

We denoted $label(a^j)$ is the set of labels of the vertices in the column vector a^j, called $B_i = \{v \in RV_i \mid label(\mathrm{v}) \in label(\mathrm{a}^j)\}$ is the set of unalign vertices of the graph G_i (denote by RV_i) whose labels are like to the labels of the vertices in the alignment vector a^j. In the case of having no vertices which have label belong to *label*(a^j). B_i will be assigned by the set of unalign remaining vertices. Ant will randomly select a node in B_i with the probability given in Equation 4.

For ease of visualization, we assume the ant start from the graph G_1 and random walk along the path $< a_1^j, a_2^j, ..., a_{i-1}^j >$ to graph G_i where it chose vertex k in G_i with probability:

$$p_{j,k}^i = \frac{(\tau_{j,k}^i)^\alpha * [\eta_{j,k}^i(\mathrm{a}^j)]^\beta}{\sum_{s \in B_i} (\tau_{j,s}^i)^\alpha * [\eta_{j,s}^i(\mathrm{a}^j)]^\beta} \qquad (4)$$

After a vector is fully developed into $a^j = (a_1^j, ..., a_n^j)^T$, the real vertices in vector a^j is removed from the construction graph to continue repeating the alignment procedure of ants until all vertices have already aligned.

d) Pheromone Update Rule

Pheromone trail intensity $\tau_{j,k}^i$ is initialized as τ_{max} and will be updated after each iteration.

After the ants found the solutions or carried out local search (in the second stage), the pheromone trail is updated according to SMMAS pheromone trail update rule in [28], [29], as follows:

$$\tau_{j,k}^i = (1-\rho)\tau_{j,k}^i + \Delta_{j,k}^i \qquad (5)$$

$$\Delta_{j,k}^i = \begin{cases} \rho * \tau_{max} & (i,j,k) \in \textit{gbest solution} \\ \rho * \tau_{mid} & (i,j,k) \in \textit{ibest solution} \\ \rho * \tau_{min} & \textit{otherwise} \end{cases} \qquad (6)$$

where τ_{max}, τ_{min} and $\rho \in (0,1)$ are given parameters, best solution is the best solution found in current iteration.

Note that in Equation (5), parameter ρ defines two properties: reinforcement search around the best-found solution and explore new solution. In ACOTS-MGA, at the first stage, the ρ is set small to efficient use reinforcement information, and set it higher at the second stage to emphasise on exploration.

Focusing on equation 6, difference to ACO-MGA and ACO-MGA2, ACOTS-MGA uses combine *ibest* solution and *gbest* solution to update pheromone trail.

e) Tabu search procedure

In the last iterations of ACOTS-MGA algorithm, Tabu Search algorithm is applied to enhance the solution quality.

Tabu search procedure will review the vertices of graphs, with each graph it swap the pairs of vertices belong this graph on the alignment vectors. If this change increases the score, the best solution will be updated with the current solution. Unlike conventional search procedures, Tabu Search procedure uses a Tabu list to save the node swap. These node pairs in Tabu list will not be reviewed again to avoid being repeated the swapping of two node.

Another difference of ACOTS-MGA from the ACOMGA2 algorithm is that the local search procedure of ACOMGA2 is only called once time at each iteration, in the ACOTS-MGA algorithm, the Tabu search procedure is repeatedly called until it does not improve the solution quality anymore.

3.2. General framework

The ACOTS-MGA algorithm is implemented in multiple loops until it satisfies the predefined stop condition. It includes two stages as in Algorithm 1.

At the first 80% of iterations, in each iteration, each ant builds solutions on the construction graph based on heuristic information and pheromone trail intensity. Then the algorithm determines the best solution of the iteration, updates pheromone trail according to SMMAS rule and updates the best solution found by then.

At the last 20% of iterations, in each iteration, after ants build solutions, Tabu search techniques are applied to find the best solution of iteration. Then ACOTS-MGA updates pheromone trail according to SMMAS rule and updates the best solution.

Algorithm 1: ACOTS-MGA algorithm

```
Input: A set of graphs $G = \{G_1(V_1,E_1),\dots,G_n(V_n,E_n)\}$
Output: The best alignment
 $A \subseteq (V_1 \cup \{\perp\}) \times \dots \times (V_n \cup \{\perp\})$ for G

Begin
  Initialize; //initialize pheromone trail matrix and $n_{ant}$ ants;
  while (stop conditions not satisfied) do
          for i=1 to $n_{ant}$ do
              $ant_i$ builds a multiple graph alignment;
          Tabu search //run only at the second stage
          Update pheromone trail;
          Update the best solution;
  End while;
  Save the best solution;
End;
```

4. Experiment results

4.1. Data descriptions

The experiment data contains 74 structures extracted from Cavbase database[19]. Each structure represents a protein cavity belonging to protein family of thermolysin, bacteria protease commonly used in analysis of protein and annotated with the EC number 3.4.24.27 in the ENZYME database [8].

In this data set, each generated graph has 42 to 94 vertices. The graphs are selected from 74 structures to generate random data sets contain 4, 8, 16, 32 graphs.

4.2. Parameters and computer configuration

Because the ACO-MGA2 is an improved version of ACO-MGA, experiments presented here only compare ACOTS-MGA with Greedy [21], GAVEO [8] and ACO-MGA2[27].

The parameters of ACOTS-MGA areset as follow:

- The number of ants at each iteration is 30
- ρ_1=0.3, ρ_2=0.7 ($\rho=\rho_1$ at the first stage, and ($\rho=\rho_2$ at the second stage)
- $\alpha = \beta = 1$
- τ_{max} = 1, τ_{mid}=0.8 and $\tau_{min} = \frac{\tau_{max}}{V_{max}^2}$, where $V_{max} = max(|V_1|,|V_2|,\dots,|V_n|)$.
- Local search procedure is applied in the last 20% of iterations.

Our experiments are performed on a computer with following configuration: CPU Intel Core 2 Duo 3 Ghz, RAM DDR3 4GB and Windows 7 operating system.

4.3. Effect and runtime comparison

In this experiment, we run the algorithms on the same data sets with a predetermined number of iterations. To compare the solution quality and runtime of algorithms, we performed each algorithm on each data set 20 times and took the average values for comparison.

The score and the runtime of the algorithms are shown in Table 1 and Table 2. The experimental results in Table 1 show that ACOTS-MGA algorithm in any case has better solution quality than GAVEO and ACO-MGA2 and greedy. Especially when increasing the number of graphs, the outperformance of ACOTS-MGA over other methods is more prominent.

When comparing in terms of runtime, table 2 shows that the ACOTS-MGA algorithm run faster than the GAVEO and ACO-MGA2 does in case of the number of graphs is 4 or 8. However, in case of the number of graph is 16, ACOTS-MGA is faster than GAVEO and slower than ACO-MGA2; in case of the number of graph is 32, ACOTS-MGA is slower than ACO-MGA2 and GAVEO.

4.4. Comparing GAVEO and ACOTS-MGA under a predetermined amount of time

Because the greedy method requires small runtime and its solution quality is too bad, in this section, we only compare the solution quality of GAVEO, ACO-MGA2 and the solution quality of ACOTS-MGA in the same runtime.

We run GAVEO, ACO-MGA2 and ACOTS-MGA algorithms on a data set of 16 graphs, each graph contains 42 to 94 vertices, with the runtime increase from 1000s to the 6000s. The results are shown in Figure 3. It shows that when the runtime increases from 1000s to 6000s, solution quality of ACOTS-MGA is always better than GAVEO and ACO-MGA2 algorithm.

In addition, to compare the solution quality of ACOTS-MGA with ACO-MGA2 and GAVEO algorithms in the same time. We run the GAVEO and ACO-MGA2 algorithm on the same dataset at the same time as the runtime of the ACOTS-MGA algorithm given in Table 1. The results are shown in table 3. It can be seen from table 3 that when running in the same time, with all data sets, ACOTS-MGA algorithm is better than ACO-MGA2 and GAVEO.

Table 1. Comparison of the score of algorithms with the data sets consisting of 4, 8, 16 and 32 graphs

Method/Number of graphs	**4**	**8**	**16**	**32**
Greedy	-4098.00	-11827.00	-56861.00	-267004.00
GAVEO	-1223.50	-2729.67	-10604.00	-63205.33
ACO-MGA2	-971.80	-2277.80	-7857.20	-53960.10
ACOTS-MGA	**-963.12**	**-1088.81**	**-5670.86**	**-42215.91**

Table 2. Comparison of the algorithm runtime (seconds) with the data sets consisting of 4, 8, 16 and 32 graphs

Method/Number of graphs	**4**	**8**	**16**	**32**
GAVEO	1892 s	2851 s	10067 s	20671 s
ACO-MGA2	272 s	1374 s	**4151 s**	**18005 s**
ACOTS-MGA	**171 s**	**809 s**	6839 s	53800 s

Table 3. Comparison of score of GAVEO, ACO-MGA2 and ACOTS-MGA algorithms with the same runtime with datasets include 4,8,16 and 32 graphs

Method/Number of graphs	**4**	**8**	**16**	**32**
GAVEO	-1223.50	-2879.00	-10744.00	-63205.33
ACO-MGA2	-989.48	-1524.43	-7757.20	-53960.10
ACOTS-MGA	**-963.12**	**-1088.81**	**-5670.86**	**-42215.91**

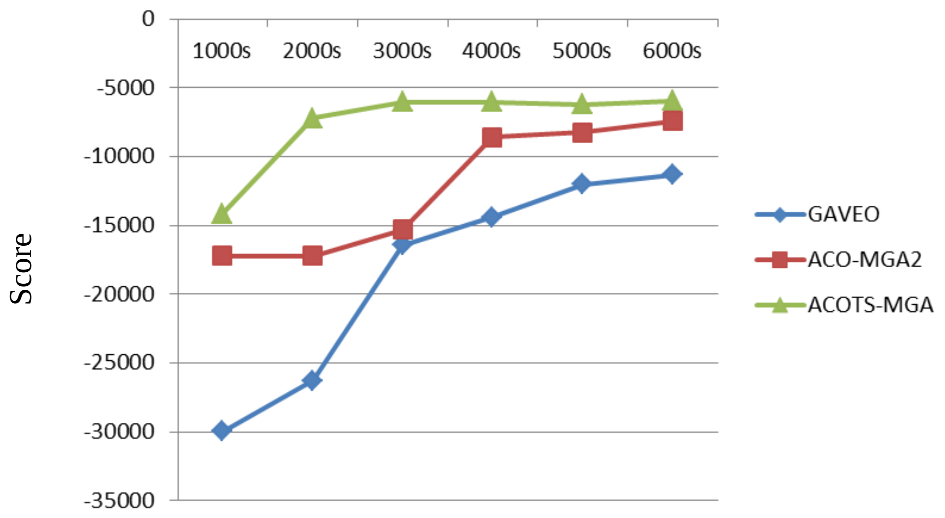


Figure 3. Comparison of results of ACOTS-MGA algorithm with ACO-MGA2 and GAVEO algorithms with data set of 16 graphs when runtime increase from 1000s to 6000s.

5. Conclusions

This paper proposes a new algorithm for solving a multi-graph alignment problem called ACOTS-MGA. This algorithm is an improvement of the ACO-MGA2 algorithm. In ACOTS-MGA, the local search procedure is replaced by Tabu Search procedure. In addition, there are some changes in ACOTS-MGA: the random walk procedure to construct the solution, heuristic information and pheromone update manner. Experiments on the real data set show that the proposed algorithm yield the solution quality better than previous algorithms.

When the number of graphs increases, the proposed algorithm runs slowly. However, as well as the other ACO-based algorithms, ACOTS-MGA could be implemented as parallel to work with the higher number of graphs.

References

[1] A. E. Todd, C. A. Orengo, and J. M. Thornton, "Evolution of function in protein superfamilies, from a structural perspective," *J. Mol. Biol.*, vol. 307, no. 4, pp. 1113–1143, Apr. 2001.

[2] S. F. Altschul *et al.*, "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs," *Nucleic Acids Res.*, vol. 25, pp. 3389–3402, 1997.

[3] R. C. Edgar, "MUSCLE: multiple sequence alignment with high accuracy and high throughput," *Nucleic Acids Res.*, vol. 32, no. 5, pp. 1792–1797, Mar. 2004.

[4] 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 Res.*, vol. 22, no. 22, pp. 4673–4680, Nov. 1994.

[5] M. Larkin, G. Blackshields, N. Brown, … R. C.-, and undefined 2007, "Clustal W and Clustal X version 2.0," *academic.oup.com*.

[6] C. Notredame, D. G. Higgins, and J. Heringa, "T-coffee: a novel method for fast and accurate multiple sequence alignment," *J. Mol. Biol.*, vol. 302, no. 1, pp. 205–217, Sep. 2000.

[7] K. Sjolander, "Phylogenomic inference of protein molecular function: advances and challenges," *Bioinformatics*, vol. 20, no. 2, pp. 170–179, Jan. 2004.

[8] T. Fober, M. Mernberger, G. Klebe, and E. Hüllermeier, "Evolutionary construction of multiple graph alignments for the structural analysis of biomolecules," *Bioinformatics*, vol. 25, no. 16, pp. 2110–2117, 2009.

[9] M. Mernberger, G. Klebe, and E. Hullermeier, "SEGA: Semiglobal Graph Alignment for Structure-Based Protein Comparison," *IEEE/ACM Trans. Comput. Biol. Bioinforma.*, vol. 8, no. 5, pp. 1330–1343, Sep. 2011.

[10] D. Shasha, J. T. L. Wang, and R. Giugno, "Algorithmics and applications of tree and graph

searching," in *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems - PODS '02*, 2002, p. 39.

[11] R. V. Spriggs, P. J. Artymiuk, and P. Willett, "Searching for Patterns of Amino Acids in 3D Protein Structures," *J. Chem. Inf. Comput. Sci.*, vol. 43, no. 2, pp. 412–421, Mar. 2003.

[12] D. Conte, P. Foggia, C. Sansone, And M. Vento, "Thirty years of graph matching in pattern recognition," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 18, no. 3, pp. 265–298, May 2004.

[13] K. Kinoshita and H. Nakamura, "Identification of the ligand binding sites on the molecular surface of proteins," *Protein Sci.*, vol. 14, no. 3, pp. 711–718, Mar. 2005.

[14] O. Kuchaiev and N. Pržulj, "Integrative network alignment reveals large regions of global network similarity in yeast and human," *Bioinformatics*, vol. 27, 2011.

[15] Xifeng Yan, Feida Zhu, Jiawei Han, and P. S. Yu, "Searching Substructures with Superimposed Distance," in *22nd International Conference on Data Engineering (ICDE'06)*, 2006, pp. 88–88.

[16] X. Yan, P. S. Yu, and J. Han, "Substructure similarity search in graph databases," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data - SIGMOD '05*, 2005, p. 766.

[17] S. Zhang, M. Hu, and J. Yang, "TreePi: A Novel Graph Indexing Method," in *2007 IEEE 23rd International Conference on Data Engineering*, 2007, pp. 966–975.

[18] A. E. Aladag and C. Erten, "SPINAL: scalable protein interaction network alignment," *Bioinformatics*, vol. 29, pp. 917–924, 2013.

[19] S. Schmitt, D. Kuhn, and G. Klebe, "A New Method to Detect Related Function Among Proteins Independent of Sequence and Fold Homology," *J. Mol. Biol.*, vol. 323, no. 2, pp. 387–406, Oct. 2002.

[20] M. Hendlich, A. Bergner, J. Günther, and G. Klebe, "Relibase: Design and Development of a Database for Comprehensive Analysis of Protein–Ligand Interactions," *J. Mol. Biol.*, vol. 326, no. 2, pp. 607–620, Feb. 2003.

[21] N. Weskamp, E. Hüllermeier, D. Kuhn, and G. Klebe, "Multiple graph alignment for the structural analysis of protein active sites," *IEEE/ACM Trans. Comput. Biol. Bioinforma.*, vol. 4, no. 2, pp. 310–320, 2007.

[22] T. N. Ha, D. D. Dong, and H. X. Huan, "An efficient ant colony optimization algorithm for Multiple Graph Alignment," in *2013 International Conference on Computing, Management and Telecommunications (ComManTel)*, 2013, pp. 386–391.

[23] F. Neri, *Handbook of memetic algorithms*, vol. 379. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[24] M. Gong, Z. Peng, L. Ma, and J. Huang, "Global Biological Network Alignment by Using Efficient Memetic Algorithm," *IEEE/ACM Trans. Comput. Biol. Bioinforma.*, vol. 13, no. 6, pp. 1117–1129, Nov. 2016.

[25] J. M. Caldonazzo Garbelini, A. Y. Kashiwabara, and D. S. Sanches, "Sequence motif finder using memetic algorithm," *BMC Bioinformatics*, vol. 19, 2018.

[26] L. Correa, B. Borguesan, C. Farfan, M. Inostroza-Ponta, and M. Dorn, "A Memetic Algorithm for 3-D Protein Structure Prediction Problem," *IEEE/ACM Trans. Comput. Biol. Bioinforma.*, pp. 1–1, 2016.

[27] H. Tran Ngoc, D. Do Duc, and H. Hoang Xuan, "A novel ant based algorithm for multiple graph alignment," in *2014 International Conference on Advanced Technologies for Communications (ATC 2014)*, 2014, pp. 181–186.

[28] H. X. Huan, N. Linh-Trung, H.-T. Huynh, and others, "Solving the Traveling Salesman Problem with Ant Colony Optimization: A Revisit and New Efficient Algorithms," *REV J. Electron. Commun.*, vol. 2, no. 3–4, 2013.

[29] D. Do Duc, H. Q. Dinh, and H. Hoang Xuan, "On the Pheromone Update Rules of Ant Colony Optimization Approaches for the Job Shop Scheduling Problem," 2008, pp. 153-160.