ISSN 2047-3338

A New Genetic Algorithm Applied to Inexact Graph Matching

Le Dang Nguyen[1], Dac-Nhuong Le[1], Tran Thi Huong[2] and Le Trong Vinh[2]
[1]Haiphong University, Haiphong, Vietnam
[2]Hanoi University of Science, Vietnam National University, Hanoi, Vietnam
nhuongld@hus.edu.vn, nguyenld@hus.edu.vn, tranthihuong@hus.edu.vn, vinhlt@hus.edu.vn

***Abstract*—Graph matching is used for model-based pattern recognition of brain images, model design objects in a computer-aided design, machine learning, data mining, packet filtering, web phishing, etc. In this paper, we have proposed a new genetic algorithm for inextract graph matching with many types of graph such as undirected, directed, weighted, and labeled. The experimental results show that our proposed algorithm has achieved a much better performance than other deterministic algorithms.**

***Index Terms*—Graph Matching and Genetic Algorithm**

I. Introduction

GRAPH theory is a powerful and useful versatile tool in various subfields of computer science and computer network such as: pattern recognition, scene analysis, chemistry, molecular biology, computer science, etc. An important problem in many applications is to find similarities between objects. If we use a graph-based representation, the problem turns into finding similarities between graphs, which includes tasks as exact and inexact matching, where the graph/subgraph isomorphism detection is a critical operation. This is also known as graph matching. Graph matching is used for model-based pattern recognition of brain images, for model design objects in a computer-aided design, machine learning, data mining, packet filtering and web phishing, etc.

In general, assume that, there are two graphs, the model graph G_M and the data graph G_D – the procedure of comparing them involves to check whether they are similar or not. We can define the problem of graph matching as follows: Given two graphs $G_M = (V_M, E_M)$ and $G_D = (V_D, E_D)$, with $|V_M| = |V_D|$, the problem is to find a one-to-one mapping $f : V_M \rightarrow V_D$ such that $(u,v) \in E_M$ iff $(f(u), f(v)) \in E_D$. When such a mapping f exists, this is called an isomorphism, and G_D is said to be isomorphic to G_M. This type of problems is said to be the exact graph matching. The term inexact applied to some graph matching problems means that it is not possible to find an isomorphism between the two graphs to be matched. This is the case when the number of vertices is different in both the model and data graphs. This may be due to the schematic aspect of the model and the difficulty to segment accurately the image into meaningful entities. Therefore, in these cases no isomorphism can be expected between both graphs, and the graph matching problem does not consist in searching for the exact way of matching vertices of a graph with vertices of the other, but in finding the best matching between them. This leads to a class of problems known as inexact graph matching. In that case, the matching aims at finding a non-objective correspondence between a data graph and a model graph. In the following we will assume $|V_M| < |V_D|$. The complexity of inexact graph matching is proved in [1] to be NP-complete.

In this paper, we propose a new genetic algorithm to solve inexact graph matching problems. The rest of this paper is organized as follows. Section II presents related works. Section III presents our new algorithm for inexact graph matching. Section IV presents our simulation and analysis results, and finally, Section V concludes the paper.

II. Related Works

A wide spectrum of graph matching algorithms with different characteristics have been made available recently. The standard algorithm for graph and subgraph isomorphism detection is the one by Ullman [2]. Maximum common subgraph detection has been addressed in [3], [4], [5]. Classical methods for error-tolerant graph matching can be found in [6]-[10]. Most of these algorithms are particular versions of the A* search procedure, i.e., they rely on some kind of tree search incorporating various heuristic look ahead techniques in order to prune the search space [11]. These methods are guaranteed to find the optimal solution but require exponential time and space due to the NP-Completeness of the problem. Sub-optimal, or approximated methods, on the other hand, are polynomially bounded in the number of computation steps but may fail to find the optimal solution. In [12], [13] a new method is described for matching a graph G against a database of model graphs $G_1,\ldots,G_n$ in order to find the model G_i with the smallest edit distance $d(G,$

G_i) to G and the models in the database are not completely dissimilar. The model graphs $G_1,\ldots,G_n$ are preprocessed generating a symbolic data structure, called network of models. This network is a compact representation of the models in the sense that multiple occurrences of the same subgraph s_j are represented only once. Consequently, such subgraphs will be matched only once with the input. Hence the computational effort will be reduced. In [14], an even faster algorithm for graph and subgraph isomorphism detection was introduced. It is based on an intensive preprocessing step in which a database of model graphs is converted into a decision tree. At run time, the input graph is classified by the decision tree and all model graphs for which there exists a subgraph isomorphism from the input are detected. For example, we have two directed and labeled graphs G_M and G_D shown in Fig. 1.

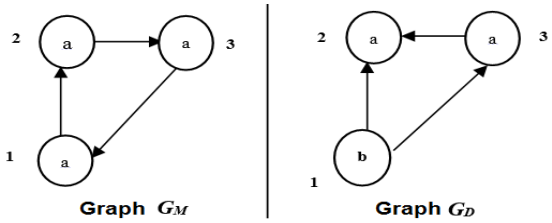
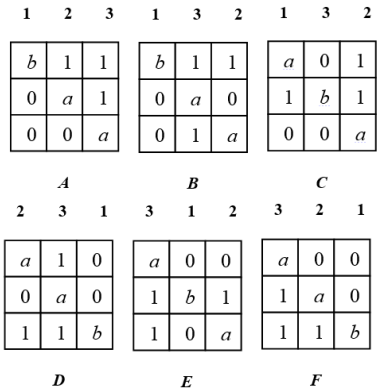


Figure 1. An example of graph matching

We represent G_M and G_D with two adjacency matrices M_1 and M_2. Then, we find all possible permutation matrices of M_1, M_2, presented in a decision tree as in Fig. 2.

The interest of inexact graph matching has been increased in the recent years. Many algorithms have been proposed for inexact graph matching. We can find in the literature various techniques applying probability theory to graph matching problems. A review on general purpose probabilistic graph matching with different types of probabilistic graphs, different techniques for their manipulation, and fitness functions appropriated to use for these problems are presented by Bengoetxea et al in [15], Cesar et al in [16], Christmas et al in [17], Coughlan et al in [18], Skomorowski in [19], Williams et al in [20], Wilson and Hancock in [21]. Probabilistic relaxation is also used for solving the graph matching problem when formulated in the Bayesian framework for contextual



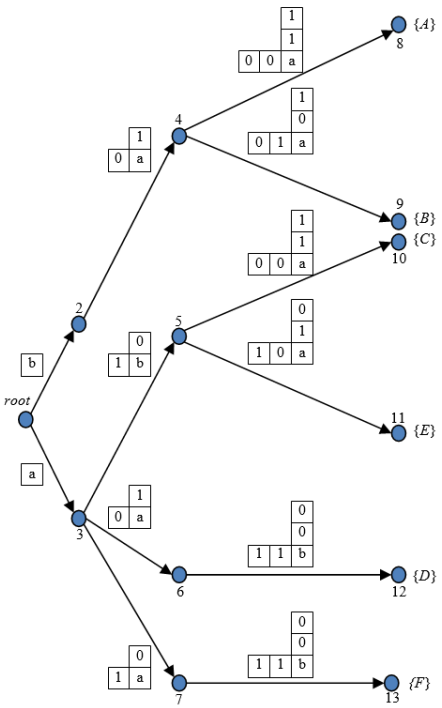


Figure 2. A decision tree representing all the adjacency matrices of the graph G_D

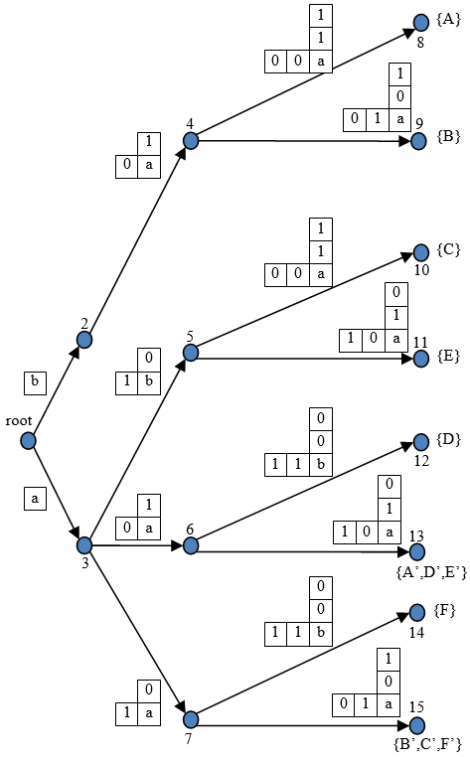


Figure 3. A decision tree representing all the adjacency matrices of the graph G_M on decision trees represening the adjacency matrices of G_D

label assignment [22], and the same idea is applied in [21] combining several popular relational distance measures and an active process of graph-editing.

Another important approach is the EM algorithm presented by Cross and Hancock in [23], Finch and et. al in [24]. Decision trees have also been applied to graph matching, Messmer and Bunke had used decision trees for solving the largest common subgraphproblem instead of applying queries to a database of models [25]. Finally, the fact of formulating complex graph matching problems as combinatorial optimization ones is not novel, and many references applying different techniques in this field can be found in the literature. Genetic algorithms proposed by Cross et al in [23], Khoo and Suganthan in [26], Singh et al in [27]. However, some optimization and modifications still need to be carried out on the algorithm in order to improve the performance and accuracy for matching similarities between two graphs.

III. A New Genetic Algorithm

In this section, we propose a new genetic algorithm to solve inexact graph matching problems with many types of graph such as: directed and indirected, attributed, weighted graphs. The best correspondence of a graph matching problem is defined as the optimum of some objective function which measures the similarity between matched vertices and edges. This objective function is also called fitness function.

Given two graphs $G_M = (V_M, E_M)$ and $G_D = (V_D, E_D)$, with $|V_M| = |V_D|$, the problem is to find a one-to-one mapping $f: V_M \rightarrow V_D$ such that $(u,v) \in E_M$ iff $(f(u), f(v)) \in E_D$. If the total number of pairs of vertices be the greater the mapping f is considered appropriate. Thus, the edge $(f(u), f(v))$ is the image of the edge (u, v) in the graph G_D. Obviously, each image of the graph G_M through the mapping f is a candidate for a graph of the number of vertices in the graph G_D by the number of vertices of the graph G_M. The total number of candidates to be considered (for some subgraphs of the graph G_D) is C_n^m (m combinations of n).

Genetic algorithms belong to a larger class of evolutionary algorithms, which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover. Our genetic algorithm is described as follows:

A. Represent and decode an individual

Each individual is a vector of n components $I=(i_1,i_2,\ldots,i_j,\ldots i_m)$ which means that we map vertex $j \in V_M$ to vertex $i_j \in V_D$ or vertex $i_j \in V_D$ is the image of the vertex $j \in V_M$. In this way, each individual I has m genes with m vertices in G_D.

B. Initialization

We use fully random initialization in order to initialize a set of individuals P. Each individual is initialized by a randomly generated integer m in $[1..n]$.

C. Fitness Function

We define a function to check for each $(u,v) \in E_M$, such that

$$check(u,v) = \begin{cases} 1 & \text{if } (u,v) \in E_M \text{ and } (f(u), f(v)) \in E_D \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

With different types of graph, the testing conditions for edge (u,v) satisfying the map f are different:

- *Undirected graph* (one of the edges has no direction. Edge (u,v) is identical to edge (v,u)): A condition is said satisfied by function f if given edge (u,v) is an edge of graph G_M then $(f(u), f(v))$ is an edge of graph G_D, where $f(u)$, $f(v)$ are images of u, v over function f.
- *Directed graph:* A condition is said satisfied by function f if given edge (u,v) is an arc of graph G_M then $(f(u), f(v))$ is an arc of graph G_D.
- *Undirected Weighted graph (Directed):* A condition is said satisfied by function f if given (u,v) is an edge (arc) of graph G_M then $(f(u), f(v))$ is an edge (arc) of graph G_D and their weights must be equal.
- *Undirected Labeled graph (Directed):* This case is similar to the case of Undirected Weighted graph (Directed). However, vertices of the two graphs are labeled the same.

The cost function $cost(I)$ calculates the total of all edges (u,v) satisfying the mapping f is given by:

$$\text{cost}(I) = \sum_{u,v \in V_M} check(u,v) \quad (2)$$

The fitness function is given by:

$$fit(I) = \frac{1}{\text{cost}(I)} \quad (3)$$

If the $cost(I)$ be the greater then $fit(I)$ is as close to 1 as possible so the solution I is as close to the optimal solution.

D. The crossover operator

This operator mimics the mating process in the nature. The edges $(u,v) \in E_M$ mapped to $(f(u),f(v)) \in E_D$ can not carry consecutive genes. Therefore, we use a multi-crossover type to derive the good parts from two individuals to obtain greater hybrid offsprings. To ensure the randomness of the crossover operator, we choose any two individuals randomly in the population with the crossover probability smaller than crossover probability p_c (p_c is a parameter of the algorithm).

E. The mutation operator

The mutation operation is a kind of random change in the individuals. In our algorithm, pointwise mutation is adopted, in which one gene in an individual is changed with a certain probability, referred to as the mutation probability. This operator allows the algorithm to search for new and more feasible individuals in new corners of the solution spaces. To do mutation, an individual is randomly selected from the individuals. First, we randomly pick an integer $k \in [1..n]$ (k is called the mutation point); then, randomly select vertex j to replace vertex k in individual I, so that vertex j does not coincide with any vertex of individual I. The mutation

operator is a random process so to ensure that the algorithm will randomly select individuals in any population *I* (individual *I* represents a subgraph of G_D,). I is randomly generated with a mutation probability of individuals. It is smaller than mutation probability p_m (p_m is a parameter of the algorithm).

F. Selection of individuals for the next generation

During each successive generation, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness-based process. In each generation, we select *P* individuals with the highest fitness value among the existing population.

G. Termination condition

The algorithm will stop after *G* generations (*G* is a parameter) or when the average value of the individuals is unchanged or we can find an individual *I* satisfying $cost(I) = |E_M|$.

H. Our algorithm

Pseudocode of our genetic algorithm for inextract graph matching problem is described as follows:

```
-------------------------------------------------------------------------------
Genetic Algorithm for Graph Matching
-------------------------------------------------------------------------------
Begin
  t=0;
  P(0)= initial_Population();
  P(0).fit=getFit(P(t));
  While(Termination condition is not satisfied)
  {
    // Crossover probability
    rd1= rand()%10/10.000;
    if (rd1<pc)
    {
        child_c = crossover(P(t));
  child_c.fit =getFit(child_c);
    }
    // Mutation probability
    rd2 = rand()%10/10.000;
  if (rd2<pm)
    {
       child_m = mutation(P(t));
       child_m.fit =getFit(child_m);
    }
    P(t+1) = bestFit(P(t), child_c, child_m);
    t++;
  }
End.
-------------------------------------------------------------------------------
```

A. The complexity of our algorithm

The complexity of our algorithm is $O(P.G.m^2.n^2)$ where, *P* is the population size, *G* is the number of iterations, *m* and n are the number of vertices of G_M and G_D respectively.

IV. Experiments And Results

A. The problems tackled

For the experiments, we used 3 types of graph which are undirected, weighted and labeled graphs. We evaluated the algorithm performance by implementing and executing the GA algorithm using C language.

For each graph class, we experimented with 3 different sizes (*less than 10 vertices, 10 to 20 vertices, and greater than 20 vertices*). Graph data templates are the standard data used in [1-3] for experimental matching algorithms in general graphs. Fig. 4 and Fig. 5 represent the graph structure of our experiments.

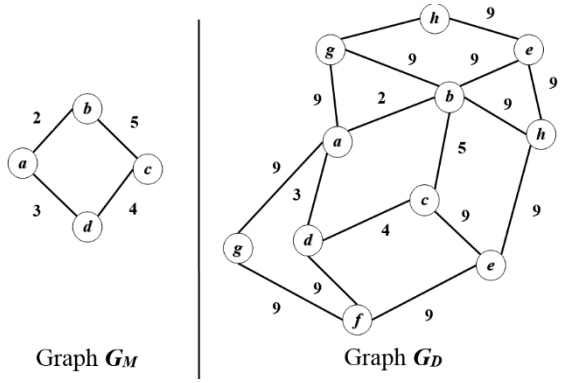


Figure 4. *Graphs G_M and G_D having 10-20 vertices*

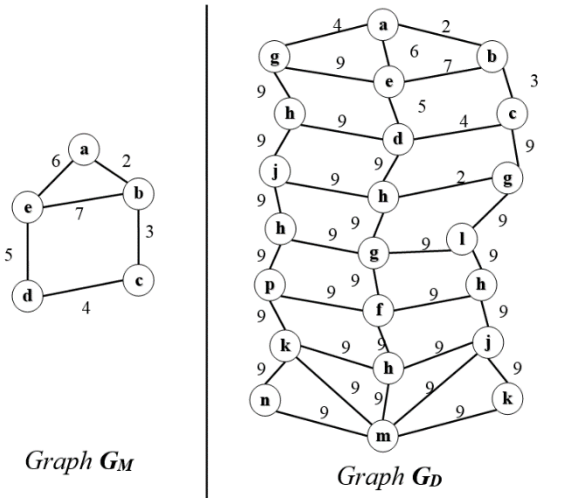


Figure 5. *The Graphs G_M and G_D having more than 20 vertices*

B. Numerical Analysis

For the graphs in Fig. 5 and Fig. 6, we remove the labels and weights to obtain the corresponding graph classes. The average results of 100 executions of our algorithm are given in Table 1 and Table 2 with the graph size greater than 10 vertices.

TABLE I. RESULT WITH GRAPHS G_M AND G_D HAVING10-20 VERTICES

Population size (*P*)	Iterations(*G*)	Optimal solution in the ith iteration
Undirected graph		
P = 8	30	11
P = 10	20	9
P = 16	15	6
P = 24	10	5
Undirected graph weighted		
P = 10	30	21
P = 16	20	13
P = 24	15	7
P = 32	10	4
Undirected graph labeled weighted		
P = 16	40	29
P = 20	30	25
P = 24	25	18
P = 32	20	15

TABLE II. RESULT OF GRAPHS G_MAND G_D HAVE MORE THAN 20 VERTICES

Population size (*P*)	Iterations (*G*)	Optimal solution in the ith iteration
Undirected graph		
P = 16	40	40
P = 20	30	27
P = 24	25	20
P = 32	20	15
Undirected graph weighted		
P = 10	45	39
P = 16	40	35
P = 24	30	20
P = 32	20	29
Undirected graph labeled weighted		
P = 16	45	35
P = 24	35	28
P = 32	30	20
P = 38	25	18

The experimental results show that genetic algorithm can be applied on various graph types. In order to illustrate the effectiveness of the genetic algorithm, we have compared its performance with deterministic hill climbing. In case the graph has less than 20 vertices, the two algorithms produce the same solutions. However when the number of vertices is greater than 20, the deterministic algorithm starts to slow down significantly; while performance change in the generic algorithm is negligible. The number of iterations *G* in the genetic algorithm depends on the matching of the graph; while the number of iterations of the Hill algorithm depends on the number of vertices.

V. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a new genetic algorithm for inextract graph matching with many types of graph such as undirected, directed, weighted, and labeled. The experimental results show that our proposed algorithm has achieved a much better performance than other deterministic algorithms. However, there are still rooms for improvement. An effective analysis of the genetic operators, the correspondence between population size, the number of generations with a sample size of graphs and graph models are our next research targets. In addition, as much of the time of genetic algorithm is spent on executing genetic operators; our new approach is to apply PSO (Particle Swarm Optimization) and ACO (Ant Colony Optimization) algorithms to solve the problem.

ACKNOWLEDGEMENT

This research is partly supported by the QG.12.21 project of Vietnam National University, Hanoi.

REFERENCES

[1] Michael, G. R., and David, J. S (2003), Computers and Intractability, A Guide to the Theory of NP-Completeness. W. H. Freeman and Company.

[2] J. R. Ullman (1976), An algorithm for subgraph isomorphism, J. Assoc. Comput. Mach. 23, pp.31-42

[3] McGregor, J. (1982). "Backtrack search algorithms and the maximal common subgraph problem", Software-Practice and Experience, Vol. 12, pp. 23–34

[4] Levi, G. (1972). "A note on the derivation of maximal common subgraphs of two directed or undirected graphs", Calcolo, V ol. 9, pp. 341–354.

[5] Pelillo, M. (1998). " A unifying framework for relational structure matching", Proc. 14th ICPR, Brisbane, 1998

[6] Eshera, M.A. and Fu, K.S. (1984). " A graph distance measure for image analysis", IEEE Trans. SMC 14, pp.398–408.

[7] Sanfeliu, A. and Fu, K.S. (1983). " A distance measure between attributed relational graphs for pattern recognition", IEEE Trans. SMC, V ol. 13, pp. 353–363.

[8] Shapiro, L.G. and Haralick, R.M. (1981). "Structural descriptions and inexact matching", IEEE Trans. P AMI, Vol. 3, pp. 504–519

[9] Tsai, W.H. and Fu, K.S. (1979). "Error-correcting isomorphisms of attributed relational graphs for pattern recognition", IEEE Trans. SMC 9, pp. 757–768

[10] Wong, E.K. (1990). "Three-dimensional object recognition by attributed graphs", In H.Bunke and A.Sanfeliu (eds.): Syntactic and Structural Pattern RecognitionTheory and Applications, pp. 381–414. World Scientific

[11] Adel HLAOUI and Shengrui WANG, A New Algorithm for Inexact Graph Matching, University Laboratories R & D program.

[12] Messmer, B.T. (1995). "Efficient graph matching algorithms for preprocessed model graphs", PhD thesis, University of Bern, Switzerland

[13] Messmer, B.T. and Bunke, H. (1998a). " A new algorithm for error tolerant subgraph isomorphism", IEEE Trans. P AMI 20, pp. 493–505.

[14] Messmer, B.T. and Bunke, H. (1999a). " A decision tree approach to graph and subgraph isomorphism detection", Pattern Recognition 32, 1999, pp. 1979-1998

[15] Bengoetxea E., Larranaga P., Bloch I., Perchant A., 2001. Image recognition with graph matching using estimation of distribution algorithms, In: Proc. Medical Image Understanding and Analysis MIUA, pp. 89–92

[16] Cesar R., Bengoetxea E., Bloch I., 2002. Inexact graph matching using stochastic optimization techniques for facial feature recognition, In: Internat. Conf. on Pattern Recognition.

[17] Christmas, W.J., Kittler, J., Petrou, M., 1995. Structural matching in computer vision using probabilistic relaxation. IEEE Trans. Pattern Anal. Machine Intell. 17 (8), 749–764.

[18] Coughlan James, Shen Huiying, 2004. Shape matching with belief propagation: Using dynamic quantization to accommodate occlusion and clutter, In: Proc. 2004 IEEE Conf. on Computer Vision and Pattern Recognition Workshop

[19] Skomorowski, M., 1999. Use of random graph parsing for scene labeling by probabilistic relaxation. Pattern Recognition Lett. 60, 649–956.

[20] Williams, M., Wilson, R.C., Hancock, E.R., 1997. Multiple graph matching with Bayesian inference. Pattern Recognition Lett. 18 (11–13), 1275–1281.

[21] Wilson, R.C., Hancock, E.R., 1999. Graph matching with hierarchical discrete relaxation. Pattern Recognition Lett. 20 (10), 1041–1052

[22] Christmas, W.J., Kittler, J., Petrou, M., 1995. Structural matching in computer vision using probabilistic relaxation. IEEE Trans. Pattern Anal. Machine Intell. 17 (8), 749–764

[23] Cross, A.D.J., Myers, R., Hancock, E.R., 2000. Convergence of a hillclimbing genetic algorithm for graph matching. Pattern Recognit. 33(11), 1863–1880.

[24] Finch, A.W., Wilson, R.C., Hancock, E.R., 1998. Symbolic graph matching with the EM algorithm. Pattern Recognit. 31 (11), 1777–1790

[25] Messmer, B.T., Bunke, H., 1999. A decision tree approach to graph and subgraph isomorphism detection. Pattern Recognit. 32, 1979–1998

[26] Khoo, K., Suganthan, P., 2002. Evaluation of genetic operators and solution representations for shape recognition by genetic algorithms. Pattern Recognition Lett. 23 (13), 1589–1597

[27] Singh, M., Chaudhury, A., Chatterjeeand, S., 1997. Matching structural shape descriptions using genetic algorithms. Pattern Recognit. 30 (9), 1451–1462.

Nguyen Dang Le received the BSc degree in computer science and the MSc degree in information technology from College of technology, Vietnam National University in Hanoi, Vietnam, in 1996 and 2005, respectively. He currently works in Haiphong University, Vietnam. His research interests include algorithm theory, network and wireless security.

Dac-Nhuong Le received the BSc degree in computer science and the MSc degree in information technology from College of Technology, Vietnam National University, Vietnam, in 2005 and 2009, respectively. He is a lecturer at the Faculty of information technology in Haiphong University, Vietnam. He is currently a Ph.D student at Hanoi University of Science,Vietnam National University. His research interests include algorithm theory, computer network and networks security.

Tran Thi Huongreceived the BSc degree in Applied Mathematics and Informatics from Hanoi University of Science, Vietnam, in 2013. She is a lecturer at the Faculty of Mathematics, Mechanics andInformatics, Hanoi University of Science, Vietnam National University. She research interests include algorithm theory, computer network.

Vinh Trong Le received the MSc degree in Information Technology from Faculty of Mathematics, Mechanics andInformatics, Hanoi University of Science, Vietnam National University in 1997, PhD degrees in Computer Science from Japan Advanced Institute of Science and Technology in 2006, respectively. He is currently Associate professor at the Faculty of Mathematics, Mechanics andInformatics, Hanoi University of Science, Vietnam National University. His research interests include algorithm theory, network and wireless security.