

# A New Graph Invariant for Graph Isomorphism: Probability Propagation Matrix\*

GOW-HSING KING AND WEN-GUEY TZENG  
*Department of Computer and Information Science*  
*National Chiao Tung University*  
*Hsinchu, Taiwan 300, R.O.C.*

The graph isomorphism problem is to determine whether two given graphs are isomorphic or not. In this paper, we present a new graph invariant, called the probability propagation matrix. By means of this graph invariant, we present a heuristic algorithm for the problem. The algorithm is easy to implement and highly parallelizable.

**Keywords:** graph isomorphism, graph invariant, probability propagation matrix, parallel computing, computational complexity

## 1. INTRODUCTION

Two graphs  $G_1 = (V_1, E_1)$  and  $G_2 = (V_2, E_2)$  are isomorphic if there exists a one-to-one mapping  $\pi$  from  $V_1$  onto  $V_2$  such that  $(x, y) \in E_1$  if and only if  $(\pi(x), \pi(y)) \in E_2$ . Applications of graph isomorphism can be found in the fields of chemistry, network theory, information retrieval, automata theory, switching theory, linguistics, computer-aided-design, etc. [1, 6, 7, 19] The graph isomorphism problem is also important in the theory of complexity because its exact complexity status is left open. It is in *NP*-complete, but it is not known whether it is polynomial time solvable, *NP*-complete, or neither. There is some evidence that it is not likely to be *NP*-complete [12, 13, 20]. For example, if it is *NP*-complete, then the polynomial hierarchy would collapse to its second level [12, 13]. On the other hand, no *NP*-complete problems have been shown to be polynomially equivalent for their decision and counting versions, but the graph isomorphism problem has been [20]. Although there are no known efficient algorithms for isomorphism testing of general graphs, the graph isomorphism problem is polynomial time solvable if graphs are restricted to trees, planar graphs, interval graphs, permutation graphs, bounded degree graphs, etc. [11, 14, 15, 17, 18, 28]

Due to the importance of this problem in applications and computational complexity theory, much effort has been devoted to the problem in the past [2, 6, 8, 10, 12, 13, 20-22, 24, 26]. There are many heuristic algorithms for this problem [4, 5, 9, 23, 25]. Most of them are based on graph invariants which preserve properties or parameters of graphs under isomorphism, such as degree sequence, distance matrix, vertex ordering, etc. In this paper, we propose a new graph invariant called the *probability propagation matrix*. The probability propagation matrix extracts information from degrees and adjacent conditions of vertices simultaneously. After some operations on the matrix, this kind of information about vertices is propagated to their adjacent vertices. Since probability propagation matrix

---

Received April 1, 1997; accepted October 27, 1997.

Communicated by Wen-Lian Hsu.

\* This research was partially supported by the National Science Council, Taiwan, R.O.C., under contract No. NSC 83-0408-E-009-021.

ces can be computed using elementary operations of vectors and matrices only, the new graph invariant can be practically used in parallel computers. The algorithm designed using this new idea is easy to implement and highly parallelizable.

## 2. THE INVARIANT

We assume hereafter that all graphs  $G = (V, E)$  are directed and strongly connected. If  $G$  is not directed, we can direct each edge  $(u, v) \in E$  into two directed edges  $\langle u, v \rangle$ , and  $\langle v, u \rangle$  to make  $G$  directed. If  $G$  is not strongly connected, we can add a new vertex  $w$  into  $V$  and for each vertex  $v \in V$ , add two directed edges  $\langle w, v \rangle$  and  $\langle v, w \rangle$  into  $E$  to make  $G$  strongly connected. The assumption that  $G$  should be directed and strongly connected makes our heuristic algorithm work better, which will be explained later.

For a graph  $G = (V, E)$  with  $V = \{1, 2, \dots, n\}$ , let  $A$  be its adjacency matrix such that  $A[i, j]$  is 1 if and only if  $\langle i, j \rangle \in E$ , where  $A[i, j]$  is the  $(i, j)$ -entry of  $A$ . Let  $d_i$  be the out-degree of vertex  $i$ . Since  $G$  is strongly connected, each vertex  $i$  of  $G$  has an out-degree of at least 1.

**Definition 2.1:** The *probability distribution matrix*  $B$  of graph  $G$  is defined as, for  $1 \leq i, j \leq n$ ,

$$B[i, j] = \frac{A[i, j]}{d_i}$$

The probability distribution matrix  $B$  is used as follows. Let vertex  $i \in V$  have  $d_i = h$  and be adjacent to vertices  $i_1, i_2, \dots, i_h$ , where the direction is from  $i$  to  $i_k$ ,  $1 \leq k \leq h$ . Suppose that moving from vertex  $i$  to the next vertex  $i_k$ ,  $1 \leq k \leq h$ , is random and equally likely. Then, the probability of moving from vertex  $i$  to vertex  $i_k$  is  $1/h$  for each  $1 \leq k \leq h$ . Therefore, the iterative power  $B^{(k)}$  of  $B$  represents the probability of moving from one vertex to another after  $k$  random walks. Since the probability of moving from one vertex to its adjacent vertices is assigned as the inverse of its out-degree, after  $k$  random walks, a vertex's out-degree and adjacency condition are propagated to the vertices that are of a distance less than or equal to  $k$  away. We can see that  $B^{(k)}[i, j]$  is the probability of moving from vertex  $i$  to vertex  $j$  after  $k$  random walks.

The intuition of our invariant is based on the idea of random walks. Given the initial position at vertex  $i \in V$ , called *the initial vertex* for the sake of abbreviation, there is a probability that it shall stop at each vertex  $j$ ,  $1 \leq j \leq n$ , for each number of random walks. If two graphs  $G_1$  and  $G_2$  are isomorphic and  $\pi$  is one of the bijective mappings that are the witnesses of isomorphism of  $G_1$  and  $G_2$ , then given the initial vertices  $i$  and  $\pi(i)$  of  $G_1$  and  $G_2$ , respectively, after any number of random walks, for each vertex  $j$  of  $G_1$  the probability that it will stop at vertex  $j$  of  $G_1$  must be equal to the probability that it will stop at vertex  $\pi(j)$  of  $G_2$ . Furthermore, if  $G_1$  and  $G_2$  are not isomorphic, then for any bijective mapping  $\pi$  between  $V_1$  and  $V_2$  and for any initial vertices  $i$  and  $\pi(i)$  of  $G_1$  and  $G_2$ , respectively, it is very likely that there is a vertex  $j$  of  $G_1$  so that the probability that it will stop at vertex  $j$  of  $G_1$  is not equal to the probability that it will stop at vertex  $\pi(j)$  of  $G_2$  after a certain number of random walks. Therefore, a necessary condition for  $G_1$  and  $G_2$  to be isomorphic is as follows. There are vertices  $i$  and  $i'$  in  $G_1$  and  $G_2$ , respectively, which are initial vertices, and

a bijective mapping  $\pi$  from  $V_1$  to  $V_2$  with  $i' = \pi(i)$  such that after any number of walks, for any two vertices  $j$  and  $j' = \pi(j)$ ,  $1 \leq j, j' \leq n$ , the probability that it will stop at vertex  $j$  of  $G_1$  is equal to the probability that it will stop at vertex  $j'$  of  $G_2$ .

However, to test the necessary condition requires verifying the equality of all the corresponding probabilities for “all numbers of walks”. Fortunately, a result in [26] for determining the equivalence of the probabilistic finite automata in polynomial time can be used to test this condition. We first define probabilistic finite automata in the following.

**Definition 2.2:** A *probabilistic finite automaton* (PA)  $U$  is a 4-tuple  $(Q, \Sigma, M, \Gamma)$  with the following:

1.  $Q = \{1, 2, 3, \dots, n\}$  is a finite set of states.
2.  $\Sigma$  is the input alphabet.
3.  $M$  is a transition function from  $\Sigma$  into  $(n \times n)$ -dimensional matrices such that  $M(\sigma)[i, j]$  is the probability that PA  $U$  moves from state  $i$  to state  $j$  after reading a symbol  $\sigma \in \Sigma$ .  $M(\sigma)$  is stochastic; that is, for any  $i$ ,  $\sum_{j=1}^n M(\sigma)[i, j] = 1$ . We extend the domain of  $M$  from  $\Sigma$  to  $\Sigma^*$  as  $M(\epsilon) = I_n$  and  $M(x\sigma) = M(x) \cdot M(\sigma)$  for  $x \in \Sigma^*$  and  $\sigma \in \Sigma$ , where  $\epsilon$  is the empty string, and  $I_n$  is the  $(n \times n)$ -dimensional identity matrix. Then,  $M(x)[i, j]$  is the probability that  $U$  will move from state  $i$  to state  $j$  after reading string  $x$ .
4.  $\Gamma = [\gamma_1, \gamma_2, \dots, \gamma_n]$  with  $\gamma_i \geq 0$  and  $\sum_{i=1}^n \gamma_i = 1$ .  $\Gamma$  is called the *initial-state distribution vector* with the  $i$ th element denoting the probability of state  $i$  being the initial state. Since the algorithm presented in this paper always chooses only one state, say state  $i$ , as the initial state, we let  $\Gamma^i$  denote the row vector with  $\gamma_i = 1$  and  $\gamma_j = 0$  if  $j \neq i$ .
5. The *state distribution vector* of  $U$  with initial state  $i$  and reading string  $x$  is

$$P_U^i(x) = \Gamma^i \cdot M(x).$$

Collecting the state distribution vectors,  $P_U^i(\epsilon)$ ,  $P_U^i(\sigma_1)$ ,  $P_U^i(\sigma_1\sigma_2), \dots$ , and  $P_U^i(\sigma_1\sigma_2 \dots \sigma_k)$  produces a *probability propagation matrix*  $P_U^i[x]$ , where  $x = \sigma_1\sigma_2 \dots \sigma_k$ .

**Definition 2.3:** For PA  $U$  with initial state  $i$  and reading string  $x = \sigma_1, \sigma_2 \dots \sigma_k$ , the *probability propagation matrix*  $P_U^i[x]$  is a collection of state distribution vectors  $P_U^i(\epsilon)$ ,  $P_U^i(\sigma_1)$ ,  $P_U^i(\sigma_1\sigma_2), \dots$ , and  $P_U^i(\sigma_1\sigma_2 \dots \sigma_k)$ ; that is,

$$P_U^i[x] = \begin{bmatrix} P_U^i(\epsilon) \\ P_U^i(\sigma_1) \\ P_U^i(\sigma_1\sigma_2) \\ \vdots \\ P_U^i(\sigma_1\sigma_2 \dots \sigma_k) \end{bmatrix}$$

A digraph  $G = (V, E)$  can be transformed to a PA  $U = (Q, \Sigma, M, \Gamma)$  according to following rules. Each vertex in  $V$  is viewed as a state in  $Q$  so that  $Q = V$ . Since edges in  $E$  are not labeled, we label them using symbol  $a$ . Thus,  $\Sigma = \{a\}$ . The transition function  $M$  is defined as  $M(a) = B$ , where  $B$  is the probability distribution matrix of  $G$ . Let  $P_U^{i,j}[a^k]$

denote the  $j$ th column vector of probability propagation matrix  $P_U^i[a^k]$ . Each state  $j$  in  $Q$  corresponds to a column vector  $P_U^{i,j}[a^k]$ . The  $l$ th element of  $P_U^{i,j}[a^k]$  is the probability that  $U$  starts at initial state  $i$ , reads string  $a^{l-1}$ , and stops at state  $j$ .

In the following, for graph  $G_1 = (V_1, E_1)$  transformed to PA  $U_1 = (Q_1, \Sigma, M_1, \Gamma_1)$ , we abuse the notation a little so that  $P_1^i[x]$  denotes the probability propagation matrix of  $U_1$  with initial state  $i$ , and  $P_1^{i,j}[x]$  is the  $j$ th column of matrix  $P_1^i[x]$ . We define the isomorphism of functions of probabilistic propagation matrices as follows.

**Definition 2.4:**  $P_1^i$  is isomorphic to  $P_2^j$  if and only if there is a permutation  $\pi$  such that  $j = \pi(i)$  and

$$\forall x \in \Sigma^*, 1 \leq k \leq n, P_1^{i,k}[x] = P_2^{j,\pi(k)}[x].$$

By a result in [27], to determine the isomorphism of  $P_1^i$  and  $P_2^j$  for all input strings, it is sufficient to test their equality for strings of length less than  $2n$ .

**Theorem 2.5:**  $P_1^i$  is isomorphic to  $P_2^j$  if and only if  $P_1^i[a^{2n-1}]$  is equal to  $P_2^j[a^{2n-1}]$ .

**Proof:** The “only if” part is obviously true.

The “if” part is shown below. Since  $P_1^i[a^{2n-1}]$  is isomorphic to  $P_2^j[a^{2n-1}]$ , there is a permutation  $\pi$  such that with  $j = \pi(i)$ ,

$$\forall 1 \leq k \leq n, P_1^{i,k}[a^{2n-1}] = P_2^{j,\pi(k)}[a^{2n-1}].$$

In [27], two PA,  $U_1$  and  $U_2$ , can be shown to be equivalent by fixing accepting states. That is,  $U_1$  and  $U_2$  are equivalent if and only if for each string  $x$ , the probabilities that  $U_1$  and  $U_2$  will enter an accepting state after reading  $x$  are equal. It can be seen that the isomorphism of  $P_1^i$  and  $P_2^j$  is stronger than the equivalence of PAs since they are equivalent for the probabilities of all states, not just the sum of the probabilities of accepting states. Furthermore, it is shown that  $U_1$  and  $U_2$  are equivalent if and only if they are equivalent for strings up to length  $2n-1$ . By fixing accepting states  $k$  and  $\pi(k)$  in  $U_1$  and  $U_2$ , respectively, we can show that

$$\forall m \geq 0, P_1^{i,k}[a^m] = P_2^{j,\pi(k)}[a^m]$$

if and only if

$$\forall m \leq 2n-1, P_1^{i,k}[a^m] = P_2^{j,\pi(k)}[a^m].$$

Therefore, by considering all possible  $k$ ,  $P_1^i$  and  $P_2^j$  are isomorphic.  $\square$

A necessary condition for two graphs to be isomorphic is to find a permutation  $\pi$  such that their corresponding probability propagation matrices are isomorphic.

**Theorem 2.6:** Graph  $G_1 = (V_1, E_1)$  is isomorphic to graph  $G_2 = (V_2, E_2)$  only if for any fixed  $i$ , there exists some  $j$ ,  $1 \leq j \leq n$ , such that  $P_1^i$  and  $P_2^j$  are isomorphic.

**Proof:** Since  $G_1$  is isomorphic to  $G_2$ , there exists a bijective mapping  $\pi$  from  $V_1$  to  $V_2$  such that  $\langle i, j \rangle \in E_1$  if and only if  $\langle \pi(i), \pi(j) \rangle \in E_2$ . Therefore,

$$B_1[i, j] = B_2[\pi(i), \pi(j)]. \quad (1)$$

Let  $U_1$  and  $U_2$  be the PAs corresponding to  $G_1$  and  $G_2$ , respectively. It is obvious that state  $i$  of the  $U_1$  mapping to state  $\pi(i)$  of  $U_2$  witnesses the equivalence of  $U_1$  and  $U_2$ .

We can show that  $\pi$  is the isomorphic mapping of  $P_1^i[a^{2n-1}]$  and  $P_2^j[a^{2n-1}]$  by induction on the length of the input string  $x$ . Let  $g_{s,t}$  and  $h_{s,t}$  be the  $(s, t)$ -entry of  $P_1^i[x]$  and  $P_2^{\pi(i)}[x]$  respectively.

1. Basis:  $|x| = 0$ . Let the initial states of  $U_1$  and  $U_2$  be  $i$  and  $\pi(i)$ , respectively. Then,

$$P_1^i(\varepsilon) = [g_{0,1}, g_{0,2}, \dots, g_{0,n}] \text{ with } g_{0,i} = 1 \text{ and } g_{0,t} = 0 \text{ if } t \neq i$$

and

$$P_2^{\pi(i)}(\varepsilon) = [h_{0,1}, h_{0,2}, \dots, h_{0,n}] \text{ with } h_{0,\pi(i)} = 1 \text{ and } h_{0,t} = 0 \text{ if } t \neq \pi(i).$$

Thus,  $P_1^i[\varepsilon]$  is isomorphic to  $P_2^{\pi(i)}[\varepsilon]$  via permutation  $\pi$ .

2. Induction hypothesis:  $|x| = k-1$ . The hypothesis is

$$\forall 1 \leq j \leq n, P_1^{i,j}[a^{k-1}] = P_2^{\pi(i),\pi(j)}[a^{k-1}].$$

By the hypothesis,

$$g_{k-1,j} = h_{k-1,\pi(j)} \text{ for } 1 \leq j \leq n.$$

3. Induction step:  $|x| = k$ . Since

$$\begin{aligned} P_1^i(a^k) &= [g_{k,1}, g_{k,2}, \dots, g_{k,n}] \\ &= P_1^i(a^{k-1}) \cdot B_1 \\ &= [g_{k-1,1}, g_{k-1,2}, \dots, g_{k-1,n}] \cdot B_1. \end{aligned}$$

Therefore, for  $1 \leq j \leq n$ ,

$$\begin{aligned} g_{k,j} &= \sum_{h=1}^n g_{k-1,h} \cdot B_1(h, j) \\ &= \sum_{h=1}^n h_{k-1,\pi(h)} \cdot B_2(\pi(h), \pi(j)) \\ &= h_{k,\pi(j)}. \end{aligned}$$

Therefore,  $P_1^i(a^k) = P_2^{\pi(i)}(a^k)$  and, thus,  $P_1^i[a^k] = P_2^{\pi(i)}[a^k]$ .  $\square$

**Corollary 2.7:** For a fixed  $i$ , if  $P_1^i[a^{2n-1}]$  is not isomorphic to  $P_2^j[a^{2n-1}]$  for any  $j$ ,  $1 \leq j \leq n$ , then graphs  $G_1$  and  $G_2$  are not isomorphic.

**Corollary 2.8:** If  $P_1^i[a^{2n-1}]$  and  $P_2^j[a^{2n-1}]$  are isomorphic via a permutation  $\pi$  with  $j = \pi(i)$ , then  $\pi$  is a possible isomorphic mapping for graphs  $G_1$  and  $G_2$ .

### 3. THE HEURISTIC ALGORITHM

The algorithm is shown in Table 1. Since the input graphs  $G_1 = (V_1, E_1)$  and  $G_2 = (V_2, E_2)$  are converted to the PAs  $U_1$  and  $U_2$ , respectively, their corresponding probability distribution matrices  $B_1$  and  $B_2$  are computed first. Also, let  $A_1$  and  $A_2$  be the adjacency matrices of  $G_1$  and  $G_2$ , respectively. By Corollary 2.7 and Corollary 2.8, our heuristic algorithm is designed as follows.

**Table 1. A heuristic algorithm for graph isomorphism.**

<p><b>Input:</b> Strongly connected digraphs <math>G_1(V_1, E_1)</math> and <math>G_2(V_2, E_2)</math>.</p> <p><b>Output:</b> Isomorphism, Non-isomorphism, or Unknown.</p> <ol style="list-style-type: none"> <li>1. Compute matrices <math>B_1</math> and <math>B_2</math>;</li> <li>2. is-Iso <math>\leftarrow</math> “Unknown”; <math>i \leftarrow 1</math>; <math>iter \leftarrow 0</math>;</li> <li>3. While <math>((i \leq n)</math> and <math>(\text{is-Iso} \neq \text{“Isomorphism”}))</math> Do</li> <li>4.     Compute <math>P_1^i[a^{2n-1}]</math>; <math>j \leftarrow 1</math>;</li> <li>5.     While <math>((j \leq n)</math> and <math>(\text{is-Iso} \neq \text{“Isomorphism”}))</math> Do</li> <li>6.         If <math>(iter = 0)</math> then Compute <math>P_2^j[a^{2n-1}]</math>;</li> <li>7.         is-Iso <math>\leftarrow</math> IsoTest(<math>P_1^i[a^{2n-1}]</math>, <math>P_2^j[a^{2n-1}]</math>);</li> <li>8.         <math>j \leftarrow j + 1</math>;</li> <li>9.     End While;</li> <li>10.    <math>i \leftarrow i + 1</math>; <math>iter \leftarrow 1</math>;</li> <li>11. End While;</li> <li>12. If is-Iso=“Isomorphism”, then output one of isomorphic mappings</li> <li>13. else if is-Iso=“Non-isomorphism”, then output “Non-isomorphism”</li> <li>14.     else output “Unknown”;</li> <li>15. End.</li> </ol>
---

1. If there exist fixed  $i$  and  $j$ ,  $1 \leq i, j \leq n$ , and  $P_1^i[a^{2n-1}]$  and  $P_2^j[a^{2n-1}]$  are isomorphic with respect to a permutation  $\pi$  such that  $\pi$  is an isomorphic mapping between  $G_1$  and  $G_2$ , then  $G_1$  and  $G_2$  are isomorphic.
2. If there exists a fixed  $i$ ,  $1 \leq i \leq n$ , such that  $P_1^i[a^{2n-1}]$  and  $P_2^j[a^{2n-1}]$  are not isomorphic for all  $j = 1, 2, \dots, n$ , then the algorithm reports that  $G_1$  and  $G_2$  are non-isomorphic.
3. For each  $i$ ,  $1 \leq i \leq n$ , if there exists  $j$ ,  $1 \leq j \leq n$ , such that  $P_1^i[a^{2n-1}]$  and  $P_2^j[a^{2n-1}]$  are isomorphic via a permutation  $\pi$  but  $G_1$  and  $G_2$  are not isomorphic via  $\pi$ , then the algorithm reports “unknown”, which denotes that we do not know whether  $G_1$  and  $G_2$  are isomorphic or not.

Since computing probability propagation matrices is the most time consuming part of the algorithm, the algorithm is designed to compute a few probability propagation matrices as possible. The variable *iter* is used to control  $P_2^j[a^k]$ ,  $1 \leq j \leq n$ , at most once. The procedure IsoTest in the algorithm is shown in Table 2. The procedure tests whether  $P_1^i[a^{2n-1}]$  and  $P_2^j[a^{2n-1}]$  are isomorphic. If the column vectors of  $P_1^i[a^{2n-1}]$  are sorted to  $P_1'$  via permutation  $\eta$  and the column vectors of  $P_2^j[a^{2n-1}]$  are sorted to  $P_2'$  by lexicographical order via permutation  $\omega$  such that  $P_1' = P_2'$ , then  $P_1^i[a^{2n-1}]$  and  $P_2^j[a^{2n-1}]$  are isomorphic. Then, we can test whether graphs  $G_1$  and  $G_2$  are isomorphic via  $\omega\eta^{-1}$ .

**Table 2. The procedure IsoTest.**

<p><b>Input:</b> Probability propagation matrices <math>P_1</math> and <math>P_2</math>.</p> <p><b>Output:</b> Isomorphic mapping, Non-isomorphism, or Unknown</p> <ol style="list-style-type: none"> <li>Sort column vectors of <math>P_1</math> in lexicographical order to <math>P_1'</math> and let <math>\eta</math> be the mapping of columns of <math>P_1</math> and <math>P_1'</math>;</li> <li>Sort column vectors of <math>P_2</math> in lexicographical order to <math>P_2'</math> and let <math>\omega</math> be the mapping of columns of <math>P_2</math> and <math>P_2'</math>;</li> <li>If (<math>P_1' \neq P_2'</math>), then return "Non-isomorphism" else if (<math>\forall 1 \leq i, j \leq n, A_1[\eta(i), \eta(j)] = A_2[\omega(i), \omega(j)]</math>), then return "Isomorphism" and isomorphic mapping <math>\omega\eta^{-1}</math>; else if (<math>\forall 1 \leq i, j \leq n, A_1[\eta(i), \eta(j)] \neq A_2[\omega(i), \omega(j)]</math>), then return "Unknown".</li> </ol>
---

#### Complexity analysis:

- Given  $P_1^i(x)$  and  $M_1(\sigma)$ , computing a state distribution vector  $P_1^i(x\sigma)$  requires time  $O(n^2)$  since it is the product of an  $n$ -dimensional vector and an  $(n \times n)$ -dimensional matrix. Thus, to compute a probability propagation matrix  $P_1^i[a^k]$  requires time  $O(kn^2)$ . Computing  $P_1^i[a^k]$  and  $P_2^j[a^k]$  for all  $i, j = 1, 2, \dots, n$  requires time  $O(kn^3)$ .
- In the procedure IsoTest, to sort columns of  $P_1[a^k]$  to  $P_1'[a^k]$  in lexicographical order requires time  $O(kn \log n)$  since there are  $n$  columns of  $k$  entries to sort.
- It is sufficient to choose  $k = 2n-1$  by Corollary 2.7 and Corollary 2.8 so that the degree and adjacency conditions of each vertex are propagated to other vertices. In the worst case, computing all  $P_1^i[a^k]$  and  $P_2^j[a^k]$ ,  $1 \leq i, j \leq n$ , requires time  $O(n^4)$ . Since the procedure IsoTest is executed at most  $n^2$  times, the runtime of this procedure needs time  $O(n^4 \log n)$ . Therefore, the total runtime of the algorithm is  $O(n^4 \log n)$ .
- Since there are at most  $2n^2$  probability propagation matrices to be stored, the space complexity is  $O(n^4)$ .

## 4. AN EXAMPLE

We will consider an undirected graph  $G_1 = (V_1, E_1)$ , shown in Fig. 1(a). Since  $G_1$  is a connected undirected graph, we transfer it to a strongly digraph by directing each edge  $(u, v) \in E$  into two directed edges  $\langle u, v \rangle$  and  $\langle v, u \rangle$ . Its corresponding PA  $U_1 = (Q_1, \Sigma, M_1, \Gamma_1)$ , is shown in Fig. 1(b), and the probability distribution matrix  $B_1$  is shown below.

$$B_1 = \begin{bmatrix} \frac{0}{1} & \frac{1}{3} & \frac{0}{1} & \frac{0}{1} & \frac{1}{3} & \frac{0}{1} & \frac{0}{1} & \frac{1}{3} \\ \frac{1}{3} & \frac{0}{1} & \frac{1}{3} & \frac{0}{1} & \frac{0}{1} & \frac{0}{1} & \frac{0}{1} & \frac{1}{3} \\ \frac{0}{1} & \frac{1}{3} & \frac{0}{1} & \frac{1}{3} & \frac{0}{1} & \frac{0}{1} & \frac{1}{3} & \frac{0}{1} \\ \frac{0}{1} & \frac{0}{1} & \frac{1}{3} & \frac{0}{1} & \frac{1}{3} & \frac{1}{3} & \frac{0}{1} & \frac{0}{1} \\ \frac{1}{3} & \frac{0}{1} & \frac{0}{1} & \frac{1}{3} & \frac{0}{1} & \frac{1}{3} & \frac{0}{1} & \frac{0}{1} \\ \frac{0}{1} & \frac{0}{1} & \frac{0}{1} & \frac{1}{3} & \frac{1}{3} & \frac{0}{1} & \frac{1}{3} & \frac{0}{1} \\ \frac{0}{1} & \frac{0}{1} & \frac{1}{3} & \frac{0}{1} & \frac{0}{1} & \frac{1}{3} & \frac{0}{1} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{0}{1} & \frac{0}{1} & \frac{0}{1} & \frac{0}{1} & \frac{1}{3} & \frac{0}{1} \end{bmatrix}$$

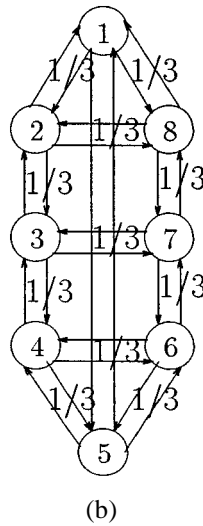
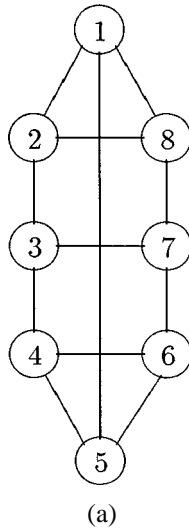


Fig. 1. Graph  $G_1$  and its corresponding PA  $U_1$ .

Suppose that state 2 of  $U_1$  is selected as the initial state. Then the initial state distribution vector  $\Gamma_1$  is

$$\Gamma_1^2 = [\frac{0}{1}, \frac{1}{1}, \frac{0}{1}, \frac{0}{1}, \frac{0}{1}, \frac{0}{1}, \frac{0}{1}, \frac{0}{1}]$$

The probability distribution vectors  $P_1^2(x)$  for some string  $x$  are computed as follows:

$$\begin{aligned}
 P_1^2(\varepsilon) &= \Gamma_1^2 \cdot M_1(\varepsilon) \\
 &= \Gamma_1^2 \cdot I_n \\
 &= \left[ \frac{0}{1}, \frac{1}{1}, \frac{0}{1}, \frac{0}{1}, \frac{0}{1}, \frac{0}{1}, \frac{0}{1}, \frac{0}{1} \right],
 \end{aligned}$$

$$\begin{aligned}
 P_1^2(a) &= \Gamma_1^2 \cdot M_1(a) \\
 &= P_1^2(\varepsilon) \cdot B_1 \\
 &= \left[ \frac{1}{3}, \frac{0}{1}, \frac{1}{3}, \frac{0}{1}, \frac{0}{1}, \frac{0}{1}, \frac{1}{3} \right],
 \end{aligned}$$

$$\begin{aligned}
 P_1^2(a^2) &= \Gamma_1^2 \cdot M_1(a^2) \\
 &= P_1^2(a) \cdot B_1 \\
 &= \left[ \frac{1}{9}, \frac{1}{3}, \frac{0}{1}, \frac{1}{9}, \frac{1}{9}, \frac{0}{1}, \frac{2}{9}, \frac{1}{9} \right],
 \end{aligned}$$

$$\begin{aligned}
 P_1^2(a^3) &= \Gamma_1^2 \cdot M_1(a^3) \\
 &= P_1^2(a^2) \cdot B_1 \\
 &= \left[ \frac{5}{27}, \frac{2}{27}, \frac{2}{9}, \frac{1}{27}, \frac{2}{27}, \frac{4}{27}, \frac{1}{27}, \frac{2}{9} \right],
 \end{aligned}$$

$$\begin{aligned}
 P_1^2(a^4) &= \Gamma_1^2 \cdot M_1(a^4) \\
 &= P_1^2(a^3) \cdot B_1 \\
 &= \left[ \frac{10}{81}, \frac{17}{81}, \frac{4}{81}, \frac{4}{27}, \frac{10}{81}, \frac{4}{81}, \frac{16}{81}, \frac{8}{81} \right], \\
 &\vdots
 \end{aligned}$$

Collecting the state distribution vectors,  $P_1^2(\varepsilon)$ ,  $P_1^2(a)$ ,  $P_1^2(a^2)$ , ..., and  $P_1^2(a^k)$ , produces a *probability propagation matrix*  $P_1^2[a^k]$ . We note here that all the states of the PA  $U$  are “distinguishable” after reading a string  $x$  if  $P_U^{i,j}[x] \neq P_U^{i,k}[x]$  for a fixed  $i$ , where  $1 \leq i, k \leq n$  and  $j \neq k$ . By “distinguishable”, we mean that all the column vectors of  $P_U^i[x]$  are different. Since all the states in  $U_1$  are “distinguishable” after PA  $U_1$  reads string  $a^3$ , we need only to compute  $P_1^2[a^3]$ , instead of  $P_1^2[a^{2n-1}] = P_1^2[a^{15}]$ . Here, we obtain

$$P_1^2[a^3] = \begin{bmatrix} \frac{0}{1} & \frac{1}{1} & \frac{0}{1} & \frac{0}{1} & \frac{0}{1} & \frac{0}{1} & \frac{0}{1} & \frac{0}{1} \\ \frac{1}{3} & \frac{0}{1} & \frac{1}{3} & \frac{0}{1} & \frac{0}{1} & \frac{0}{1} & \frac{0}{1} & \frac{1}{3} \\ \frac{1}{9} & \frac{1}{3} & \frac{0}{1} & \frac{1}{9} & \frac{1}{9} & \frac{0}{1} & \frac{2}{9} & \frac{1}{9} \\ \frac{5}{27} & \frac{2}{27} & \frac{2}{9} & \frac{1}{27} & \frac{2}{27} & \frac{4}{27} & \frac{1}{27} & \frac{2}{9} \end{bmatrix}$$

We will next consider another undirected graph  $G_2 = (V_2, E_2)$ , shown in Fig. 2(a), and test whether  $G_1$  and  $G_2$  are isomorphic. Similar to the approach taken as above, the corresponding PA  $U_2 = (Q_2, \Sigma, M_2, \Gamma_2)$ , and its probability distribution matrix  $B_2$  are computed first.  $U_2$  is shown in Fig. 2(b), and  $B_2$  is shown below:

$$B_2 = \begin{bmatrix} \frac{0}{1} & \frac{1}{3} & \frac{0}{1} & \frac{0}{1} & \frac{1}{3} & \frac{0}{1} & \frac{0}{1} & \frac{1}{3} \\ \frac{1}{3} & \frac{0}{1} & \frac{0}{1} & \frac{1}{3} & \frac{0}{1} & \frac{1}{3} & \frac{0}{1} & \frac{0}{1} \\ \frac{0}{1} & \frac{0}{1} & \frac{0}{1} & \frac{0}{1} & \frac{1}{3} & \frac{0}{1} & \frac{1}{3} & \frac{1}{3} \\ \frac{0}{1} & \frac{1}{3} & \frac{0}{1} & \frac{0}{1} & \frac{0}{1} & \frac{1}{3} & \frac{1}{3} & \frac{0}{1} \\ \frac{1}{3} & \frac{0}{1} & \frac{1}{3} & \frac{0}{1} & \frac{0}{1} & \frac{1}{3} & \frac{0}{1} & \frac{0}{1} \\ \frac{0}{1} & \frac{1}{3} & \frac{0}{1} & \frac{1}{3} & \frac{1}{3} & \frac{0}{1} & \frac{0}{1} & \frac{0}{1} \\ \frac{0}{1} & \frac{0}{1} & \frac{1}{3} & \frac{1}{3} & \frac{0}{1} & \frac{0}{1} & \frac{0}{1} & \frac{1}{3} \\ \frac{1}{3} & \frac{0}{1} & \frac{1}{3} & \frac{0}{1} & \frac{0}{1} & \frac{0}{1} & \frac{1}{3} & \frac{0}{1} \end{bmatrix}$$

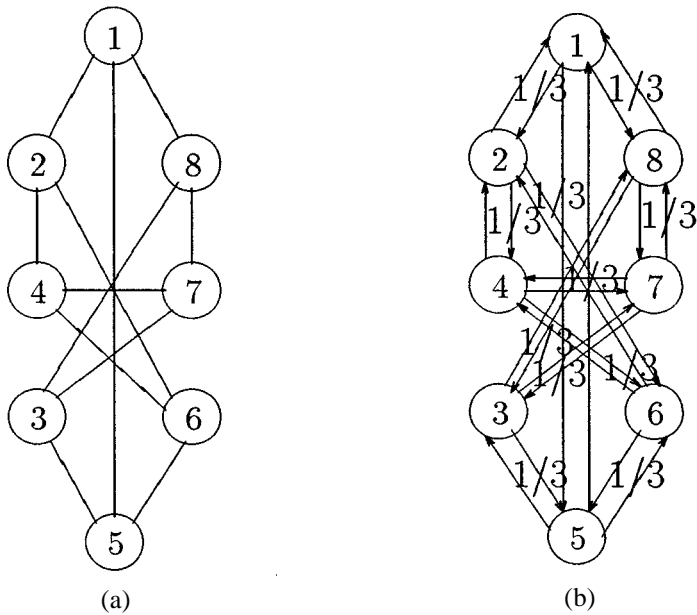


Fig. 2. Graph  $G_2$  and its corresponding PA  $U_2$ .

In this example, to test whether  $G_2$  is isomorphic to  $G_1$ , our heuristic algorithm tries to find an initial state  $i$ ,  $1 \leq i \leq n$ , to get  $P_2^i[a^3]$  such that if  $P_2^i[a^3]$  and  $P_1^j[a^3]$  are isomorphic via permutation  $\pi$ , then  $G_2$  is isomorphic to  $G_1$  via permutation  $\pi$ . On the other hand, if there is no  $j$ ,  $1 \leq j \leq n$ , such that  $P_2^i[a^3]$  and  $P_1^j[a^3]$  are isomorphic, then  $G_1$  and  $G_2$  are non-isomorphic.

Suppose that state 2 in  $U_2$  is selected as the initial state. Then, the initial state distribution vector  $\Gamma_2$  is

$$\Gamma_2 = \left[ \frac{0}{1}, \frac{1}{1}, \frac{0}{1}, \frac{0}{1}, \frac{0}{1}, \frac{0}{1}, \frac{0}{1}, \frac{0}{1} \right].$$

The probability distribution vectors  $P_2^2(x)$  for  $x = \varepsilon, a, a^2$  and  $a^3$  are as follows:

$$\begin{aligned} P_2^2(\varepsilon) &= \Gamma_2^2 \cdot M_2(\varepsilon) \\ &= \Gamma_2^2 \cdot I_n \\ &= \left[ \frac{0}{1}, \frac{1}{1}, \frac{0}{1}, \frac{0}{1}, \frac{0}{1}, \frac{0}{1}, \frac{0}{1}, \frac{0}{1} \right], \end{aligned}$$

$$\begin{aligned} P_2^2(a) &= \Gamma_2^2 \cdot M_2(a) \\ &= P_2^2(\varepsilon) \cdot B_2 \\ &= \left[ \frac{1}{3}, \frac{0}{1}, \frac{0}{1}, \frac{1}{3}, \frac{0}{1}, \frac{1}{3}, \frac{0}{1}, \frac{0}{1} \right], \end{aligned}$$

$$\begin{aligned} P_2^2(a^2) &= \Gamma_2^2 \cdot M_2(a^2) \\ &= P_2^2(a) \cdot B_2 \\ &= \left[ \frac{0}{1}, \frac{1}{3}, \frac{0}{1}, \frac{1}{9}, \frac{2}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9} \right], \end{aligned}$$

$$\begin{aligned} P_2^2(a^3) &= \Gamma_2^2 \cdot M_2(a^3) \\ &= P_2^2(a^2) \cdot B_2 \\ &= \left[ \frac{2}{9}, \frac{2}{27}, \frac{4}{27}, \frac{5}{27}, \frac{1}{27}, \frac{2}{9}, \frac{2}{27}, \frac{1}{27} \right]. \end{aligned}$$

Here, we obtain

$$P_2^2[a^3] = \begin{bmatrix} \frac{0}{1} & \frac{1}{1} & \frac{0}{1} & \frac{0}{1} & \frac{0}{1} & \frac{0}{1} & \frac{0}{1} & \frac{0}{1} \\ \frac{1}{3} & \frac{0}{1} & \frac{0}{1} & \frac{1}{3} & \frac{0}{1} & \frac{1}{3} & \frac{0}{1} & \frac{0}{1} \\ \frac{0}{1} & \frac{1}{3} & \frac{0}{1} & \frac{1}{9} & \frac{2}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{2}{9} & \frac{2}{27} & \frac{4}{27} & \frac{5}{27} & \frac{1}{27} & \frac{2}{9} & \frac{2}{27} & \frac{1}{27} \end{bmatrix}$$

We can find that  $P_1^2[a^3]$  is isomorphic to  $P_2^2[a^3]$  via the permutation

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 4 & 2 & 1 & 8 & 7 & 3 & 5 & 6 \end{pmatrix}.$$

The permutation  $\pi$  witnesses that  $G_1$  is isomorphic to  $G_2$  since

$$A_1[i, j] = A_2[\pi(i), \pi(j)], \quad 1 \leq i, j \leq n.$$

It is important to notice that if state 1 of  $G_1$  and state 1 of  $G_2$  are selected as the initial states, then there is a slight problem to in finding the isomorphic mapping since some column vectors are equal in the probability propagation matrices  $P_1^1[x]$  and  $P_2^1[x]$ , and a brute force mapping of vertices is needed.

## 5. IMPLEMENTATION AND EXPERIMENTS

Since operations on rational numbers cost too much time and space, they are not suitable for implementation. Here, we will give a modified heuristic algorithm to handle the operations.

Suppose the adjacency matrix  $A$  is the given graph  $G = (V, E)$  corresponding to  $PA$   $U$ . First, let the probability distribution matrix  $B$  be re-defined as

$$B[i, j] = A[i, j] \cdot d_i.$$

If  $d_i$  is not very large, then each entry of  $B$  can be stored as a floating point number without overflow errors, say  $d_i \ll 2^{64}$ .

Assume that the initial state of  $U$  is  $i$ . When we compute the probability propagation matrix  $P_U^i[x]$  of  $U$ , overflow errors may occur only if the length of the input string  $x$  is too long. Since vertex  $j$  and vertex  $h$  are distinct,  $P_U^{i,j}[a^k] \neq P_U^{i,h}[a^k]$  for some initial state  $i$  after reading an input string  $a^k$ . Therefore, the vertices can be partitioned into several classes after  $U$  reads some alphabets. We can sort vertices by  $P_U^{i,j}[a^k]$  lexicographically such that vertices  $j$  and  $j'$  are in the same class if  $P_U^{i,j}[a^k] = P_U^{i,j'}[a^k]$ . Let the ranking of vertex  $j$  be  $r(j)$  and  $r(j) > r(h)$  if  $P_U^{i,j}[a^k] > P_U^{i,h}[a^k]$ . The ranking is numbered from 1 to  $p$ , where  $p$  is the number of distinct classes after  $U$  reads input string  $a^k$ . We note here that all the states are “distinguishable”; if  $p = n$ , then no more input alphabets are necessary since the information is sufficient to judge whether two given graphs are isomorphic. Vertices in the same class are re-assigned the probability of the state distribution such that the last item of  $P_U^{i,j}[a^k]$  is changed to  $r(j)$ , where  $1 \leq j \leq n$ . That is, the state distribution vector  $P_U^i(a^k) = [r(1), r(2), \dots, r(n)]$ . Therefore,  $U$  can continue to read more input alphabets without overflow errors.

For example, we can consider a complete directed graph  $G = (V, E)$  of order 100. Suppose the corresponding PA of  $G$  is  $U$ . Since  $|V| = 100$ , overflow errors may result if we computes  $P_U^i[a^{199}]$  directly. Let  $U$  read input string  $a^5$  to get  $P_U^i[a^5]$  such that each entry in  $P_U^i[a^5]$  never causes any overflow errors. We can then sort  $P_U^{i,j}[a^5]$  for  $1 \leq j \leq n$  in lexicographically descending order. Since the input graph is a complete directed graph, it is obvious that all column vectors  $P_U^{i,j}[a^5]$ ,  $1 \leq j \neq i \leq n$ , are equal except  $P_U^{i,i}[a^5]$  because state  $i$  is the only initial state. Thus, state  $i$  gets rank number 1, and all the other states  $j$ ,  $1 \leq j \neq i \leq n$ , get rank number 2. Then, the new state distribution vector is  $P_U^i(a^5) = [r(1), r(2), \dots, r(n)]$ , where  $r(i)=1$  and  $r(j) = 2$ , where  $1 \leq j \neq i \leq n$ .  $P_U^i[a^k]$  should be redistributed similarly when  $k = 10, 15, \dots, 195, 199$ .

In the experiments, we randomly generated about 30,000 isomorphic and non-isomorphic pairs of undirected connected graphs with the same degree sequence from graphs of order from 10 to 1,000 to test our heuristic algorithm. In our method for random generation of graphs, if there is an edge between two vertices depends on the random number, then the average density of the tested graphs can be controlled. Regardless of the density of

the graphs, the experimental results were as follows. If the random graphs were isomorphic, our heuristic algorithm found an isomorphic mapping. If the random graphs were non-isomorphic, our heuristic algorithm reports non-isomorphism. We did not find any “unknown” cases in this random testing.

The modified heuristic algorithm was coded in ANSI C and run on a SUN SPARC 10 with 64 MB RAM and 1 MB cache. The computational time for each of the experiments is shown in Fig. 3 and Fig. 4.

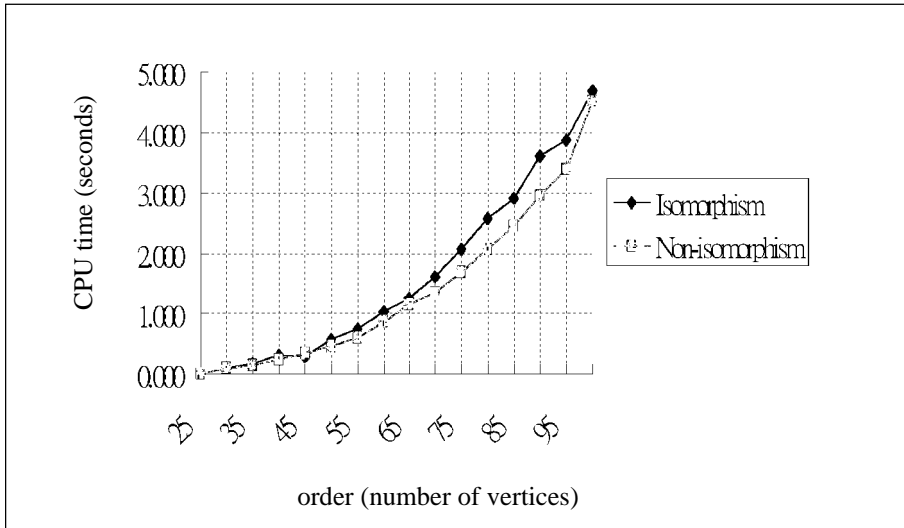


Fig. 3. Experiment result 1: order from 25 to 100.

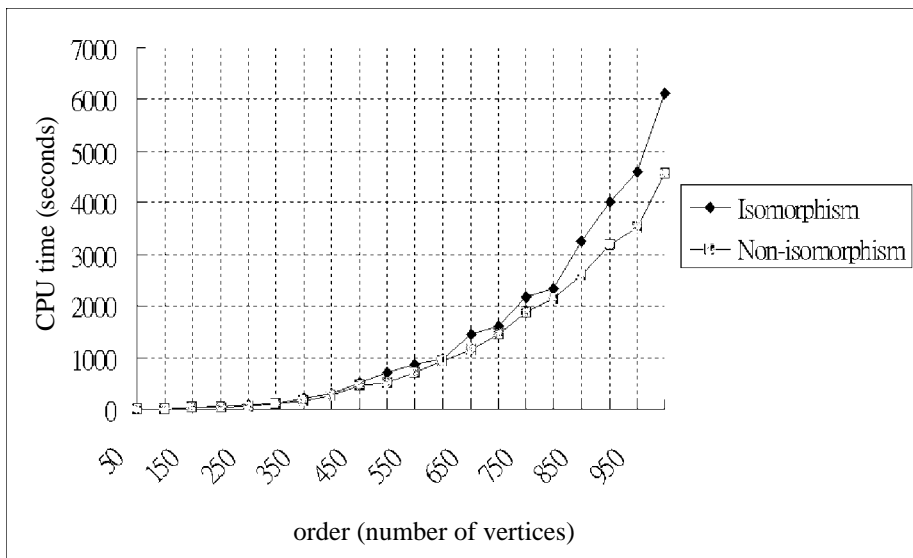


Fig. 4. Experiment result 2: order from 50 to 1000.

## 6. DISCUSSION AND CONCLUSION

Our heuristic algorithm is also applicable to testing of labelled graphs. For example, the graph representation of chemical compounds may have labelled edges and labelled vertices. If the edges in the graph are labelled, let each element  $a(i, j)$  of the adjacency matrix  $A[i, j]$  be a distinct positive integer to represent distinct labelled edges. If the vertices are labelled, the initial state distribution is defined as  $\Gamma = [c(1), c(2), \dots, c(n)]$ , where  $c(i)$  is a distinct positive integer to represent a distinct labelled vertice.

Through careful study, we find that our heuristic algorithm is weak for strongly regular graphs [3, 16, 22]; that is, if the two given graphs are strongly regular, with the same parameters, but are not isomorphic, then our algorithm outputs "unknown". Most of other heuristic algorithms [4, 9, 23, 25] are also weak for strongly regular graphs.

In this paper, we have presented an invariant for graphs. This invariant consists simultaneously of information about the adjacency conditions and degrees of vertices. Using this invariant, we have designed a heuristic algorithm for the graph isomorphism problem. We have implemented and done some experiments on this algorithm. The algorithm works quite well for randomly generated graphs.

Since the algorithm is almost involved with operations on vectors and matrices only, it is very suitable for parallel computation. Available architectures or subroutines designed for operations of vectors and matrices on parallel computers can easily and efficiently be used to implement this heuristic algorithm with less modification.

## REFERENCES

1. A.P. Ambler, H.G. Barrow, C.M. Brown, R.M. Burstall and R.J. Popplestone, "A versatile computer-controlled assembly system," in *Proceedings of Third International Joint Conference on Artificial Intelligence*, 1973, pp. 298-307.
2. L. Babai, "Isomorphism testing and symmetric of graphs," *Annals of Discrete Math*, Vol. 8, No. 1, 1980, pp. 101-109.
3. L. Babai, "On the complexity of canonical labeling of strongly regular graphs," *SIAM Journal on Computing*, Vol. 9, No. 1, 1980, pp. 212-216.
4. A.T. Bertziss, "A backtrack procedure for isomorphism of directed graphs," *Journal of ACM*, Vol. 20, No. 2, 1973, pp. 365-377.
5. K.V.S. Bhat, "Refined vertex codes and vertex partitioning methodology for graph isomorphism testing," *IEEE Transactions on System, Man & Cybernetics*, Vol. SMC-10, 1980, pp. 610-615.
6. K.S. Booth, "Isomorphism testing for graphs, semigroups, and finite automata are polynomial equivalent problems," *SIAM Journal on Computing*, Vol. 7, No. 3, 1978, pp. 273-279.
7. R. Busacker and T. Saaty, *Finite Graphs and Networks - An Introduction with Applications*, McGraw-Hill, New York, 1965, pp. 196-199.
8. M.J. Colbourn and C.J. Colbourn, "Graph isomorphism and self-complementary graphs," *SIGACT News*, Spring, 1978, pp.25-29.
9. D.G. Corneil and C.C. Gotlieb, "An efficient algorithm for graph isomorphism," *Journal of ACM*, Vol. 17, No. 1, 1970, pp. 51-64.

10. D.G. Corneil and D.G. Kirkpatrick, "A theoretical analysis of various heuristics for the graph isomorphism problem," *SIAM Journal of Computing*, Vol. 9, No. 2, 1980, pp. 281-297.
11. Z. Galil, C.M. Hoffmann, E.M. Luks, C.P. Schnorr and A. Weber, "An  $O(n^3 \log n)$  deterministic and  $O(n^3)$  Las Vegas isomorphism test for trivalent graphs," *Journal of ACM*, Vol. 34, No. 3, 1987, pp. 513-531.
12. O. Goldreich, S. Micali and A. Wigderson, "Proof that yield nothing but their validity and a methodology of cryptographic protocol design," in *27th Annual Symposium on Foundations of Computer Science*, 1986, pp.174-187.
13. S. Goldwasser and M. Sipser, "Private coins versus public coins in interactive proof systems," in *Proceedings of 18th Annual ACM Symposium Theory of Computing*, 1986, pp. 59-68.
14. J.E. Hoffmann and of J.K. Wong, "Linear time algorithm for isomorphism of planar graphs," in *Proceedings of 6th Annual ACM Symposium Theory of Computing*, 1974, pp. 172-184.
15. W.L. Hsu, "O(mn) algorithms for the recognition and isomorphism problems on circular-arc graphs," *SIAM Journal of Computing*, Vol. 24, No. 3, 1995, pp. 411-439.
16. X.L. Hubaut, "Strongly regular graphs," *Discrete Mathematics*, Vol. 13, No. 3, 1975, pp. 357-381.
17. G.S. Lueker and K.S. Booth, "A linear time algorithm for deciding interval graph isomorphism," *Journal of ACM*, Vol. 16, No. 1, 1979, pp. 183-195.
18. E.M. Luks, "Isomorphism of graphs of bounded valance can be tested in polynomial time," *Journal of Computer and System Sciences*, Vol. 25, No. 1, 1982, pp. 42-65
19. M.F. Lynch, "Storage and retrieval of information on chemical structures by computer," *Endeavour*, Vol. 27, No. 2, 1968, pp. 68-73.
20. R. Mathon, "A note on the graph isomorphism counting problem," *Information Processing Letters*, Vol. 8, No. 3, 1979, pp. 131-132.
21. G.L. Miller, "Graph isomorphism, general remarks," in *Proceedings of 9th Annual ACM Symposium Theory of Computing*, 1977, pp.143-150.
22. G.L. Miller, "On the  $n^{\log n}$  isomorphism technique," in *Proceedings of 10th Annual ACM Symposium Theory of Computing*, 1978, pp. 51-58.
23. H.B. Mittal, "A fast backtrack algorithm for graph isomorphism," *Information Processing Letters*, Vol. 29, No. 2, 1988, pp.105-110.
24. R.C. Read and D.G. Corneil, "The graph isomorphism disease," *Journal of Graph Theory*, Vol. 1, No. 4, 1977, pp. 339-363.
25. D.C. Schmidt and L.E. Druffel, "A fast backtracking algorithm to test directed graphs for isomorphism using distance matrix," *Journal of ACM*, Vol. 23, No. 2, 1976, pp. 433-445.
26. J. Shawe-Taylor and T. Pisanski, "Homomorphism of 2-complexes is graph isomorphism complete," *SIAM Journal of Computing*, Vol. 23, No. 1, 1994, pp.120-132.
27. W.G. Tzeng, "A polynomial-time algorithm for the equivalence of probabilistic automata," *SIAM Journal of Computing*, Vol. 21, No. 2, 1992, pp. 216-227.
28. S.H. Unger, "GIT - A heuristic program for testing pair of directed line graphs for isomorphism," *Communication of ACM*, Vol. 7, No. 1, 1964, pp. 26-34.



**Gow-Hsing King (金國興)** was born on Dec. 8, 1969. He received his B.S. and Ph.D. degrees in 1992 and 1998, respectively, from National Chiao Tung University. From October 1996 to January 1998, he worked as a consultant in First Global Investment Trust. He joined the Institute of Information Science, Academia Sinica, in January 1998 and has been a postdoctoral research fellow since then. Dr. King's current research interests include design and analysis of algorithms and computational finance.



**Wen-Guey Tzeng (曾文貴)** was born on March 14, 1963. He graduated from National Taiwan University in 1985. He received his Master and Ph.D. degrees in 1987 and 1991, respectively, from the State University of New York at Stony Brook. He joined the Department of Computer and Information Science, National Chiao Tung University, in 1991 and has been an Associated Professor since then. Dr. Tzeng's current research interests include cryptology, security and algorithms.