A NOTE ON THE DERIVATION OF MAXIMAL COMMON SUBGRAPHS OF TWO DIRECTED OR UNDIRECTED GRAPHS

G. Levi [1]

ABSTRACT - In this note the problem is considered of finding maximal common subgraphs of two given graphs. A technique is described by which this problem can be stated as a problem of deriving maximal compatibility classes. A known « maximal compatibility classes » algorithm can then be used to derive maximal common subgraphs. The same technique is shown to apply to the classical subgraph isomorphism problem.

1. Statement of problem.

A *graph* $G(N, E)$ consists of a non-empty set N (whose elements are called *nodes*) and a (possibly empty) set E (whose elements are called *edges*), such that $E \subset N \times N$. The *order* of G is the cardinality of set N.

A graph is *undirected*, if, for every pair of nodes $n_i, n_j\ (n_i, n_j) \in E$; it is *directed*, otherwise.

A (directed or undirected) graph G (of order n) can be represented by means of its *adjacency matrix* A, which is an $n \times n$ matrix, having the nodes of G on its rows and its columns. An element $A(i, j)$ has the value 1, if there is an edge from node n_i to node n_j; it has the value 0 otherwise. An element on the main diagonal $A(i, i)$ has the value 1 if graph G has a loop [2] on node n_i. The adjacency matrix of an undirected graph is symmetric.

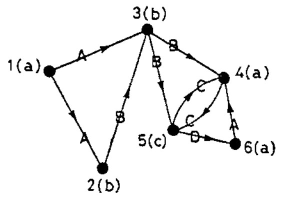
Several problems con be represented by means of *labeled graphs,* i. e. graphs in which labels are associated to nodes and/or edges. The value of

---

(1) Istituto di Elaborazione della Informazione del C.N.R. - Via S. Maria 46, 56100 Pisa.

(2) A *loop* is a pair of coincident nodes, i. e. an edge closing on itself.

the element $A(i,j)$ of the adjacency matrix is the label of edge $(n_i n_j)$, while the value of the element $A(i,i)$ is the label of node n_i. The matrix in fig. 1*b*) is the adjacency matrix of graph shown in fig. 1*a*).



a)

a	A	A			
	b	B			
		b	B	B	
			a	C	
			C	c	D
			A		a

b)

Fig. 1*a*) A labeled and directed graph: node labels are shown in brackets after the node name.
b) Adjacency matrix of graph of *a*).

Two concepts that are relevant to the present problem are the concepts of « subgraph » and « graph isomorphism ».

A *subgraph* $H(P, F)$ of $G(N, E)$ is a graph H, such that $P \subset N$ and $F = E \cap P \times P$. Subgraphs of G are obtained from G by removing some nodes and all the edges having a removed node as initial or final node.

Graph $G_1(N^1, E^1)$ is *isomorphic to* graph $G_2(N^2, E^2)$ if there exists a 1-to-1 mapping σ of N^1 onto N^2 (*isomorphism* of G_1 onto G_2), such that $(n_i^1, n_j^1) \in E^1$ if and only if $(\sigma n_i^1, \sigma n_j^1) \in E^2$, for every pair of nodes n_i^1, n_j^1 belonging to N^1. If graphs G_1 and G_2 are labeled, the further condition must hold for isomorphism that nodes and pairs of adjacent nodes (edges) of G_1 be mapped by σ onto nodes and edges of G_2 having the same labels.

The *subgraph isomorphism* problem is the following: Given two graphs G_1 and G_2, such that the order of G_1 is not greater than the order of G_2, derive all the subgraphs (if any) of G_2 which are isomorphic to G_1.

Given two graphs G_1 and G_2, if there exists a subgraph of order k G_r^1 of G_1 isomorphic to a subgraph G_s^2 of G_2, the pair of isomorphic subgraphs (G_r^1, G_s^2) is called a *common subgraph* of order k of G_1 and G_2. A common subgraph (G_r^1, G_s^2) is *maximal*, if there is no common subgraph (G_i^1, G_j^2), such that G_r^1 is a subgraph of G_i^1 and G_s^2 is a subgraph of G_j^2. If there exists a maximal common subgraph of G_1 and G_2, (G_i^1, G_j^2) of order k^*, such that $G_i^1 \equiv G_1$ (or $G_j^2 \equiv G_2$), G_1 is isomorphic to a subgraph of G_2 (G_2 is isomorphic to a subgraph of G_1). All the subgraph isomorphisms of G_1 onto G_2 (G_2 onto G_1) correspond to maximal common subgraphs of order k^*.

Deriving subgraph isomorphisms and maximal common subgraphs, is relevant in several problems, characterized by structured data (items) stored as (possibly) labeled graphs (library). Given a new item, the problem is that of testing whether it does exist in the library, possibly as a part of some known item, or whether it has some meaningful part in common with known items. Such problems arise in many different fields such as information retrieval, chemistry (chemical compounds library), artificial intelligence (question-answering systems or learning systems working on a relational data base). For example, in a chemical compounds library, the problem could be the one of testing whether a new compound is a known compound (graph isomorphism) or subcompound (subgraph isomorphism) or contains known subcompounds (common subgraphs).

Common subgraphs of order k (if any) for given graphs G_1 and G_2, of order m and n respectively, can be derived by the following two-step procedure:

i) Derive sets of subgraphs of order k G_i^1 and G_j^2 of G_1 and G_2 respectively.

ii) Test all pairs (G_i^1, G_j^2) for isomorphism.

Step ii) can be realized by a trivial algorithm, known as Node Reordering Algorithm [4], which is based on node permutations and adjacency matrix comparisons. If this algorithm is used, deriving common subgraphs of order k requires $\dfrac{m\,!\,n\,!}{k\,!\,(m-k)\,!\,(n-k)\,!}$ matrix comparisons. The efficiency of the overall algorithm cannot be substantially increased even if more efficient heuristic procedures for testing isomorphism [**1**, **2**, **4**, **5**, **6**] are used. The above mentioned procedures succeed in decreasing the number of comparisons in step ii) by computing node attributes which are invariant under isomorphism. A node n_i^1 can be mapped onto a node n_j^2 by an isomorphism only if the set of attributes of n_j^2 and n_i^1 are identical. Equality of attribute sets is a necessary condition that can be exploited in order to decrease the computation time.

A few necessary conditions for node correspondence can also be found for the subgraph isomorphism problem [see **2**, **3**]. However, taking them efficiently into account can be quite complex.

In this note a general method is devised, which is applicable to the isomorphism, subgraph isomorphism and maximal common subgraphs problems. The algorithm, while certainly not efficient in the isomorphism problem is quite convenient in the last two problems. Its main feature is that it allows to take into account necessary conditions very simply. In the next section, the only problem of maximal common subgraphs will be considered, while in section 3 properties that can be exploited in the subgraph isomorphism problem will be illustrated.

2. Node Correspondence Table, list of incompatible pairs and compatibility table.

The technique for deriving common subgraphs of given graphs $G_1(N_1, E_1)$ and $G_2(N_2, E_2)$, of order m and n respectively, is based on a table N, called *Node correspondence Table.* This table has m rows, corresponding to nodes of graph G_1 and n columns corresponding to nodes of graph G_2. An element of the table $N(i,j)$ relates to the possible mapping of i-th node of G_1, n_i^1, onto j-th node of G_2, n_j^2, and is considered to be the value of the cell (i,j). A cell (i,j), having the value 0, called *zero cell,* indicates that n_i^1 cannot be mapped onto n_j^2.

A *k-cover* of table N, is a cover of a square subtable T of N with k rows, i. e. a set S of k non-zero cells, such that there is exactly one cell of S in every row and every column of T. Subtable T defines two subgraphs of G_1 and G_2 of order k, G_T^1 and G_T^2, respectively. The k-cover can be considered a 1-to-1 mapping M_T of nodes of G_T^1 onto nodes of G_T^2. M_T is an isomorphism of G_T^1 onto G_T^2 if it maintains incidence relationships, i.e. if it maps pairs of adjacent nodes (edges) of G_T^1 onto pairs of adjacent nodes of G_T^2, and pairs of non-adjacent nodes (non-edges) of G_T^1 onto pairs of non-adjacent nodes of G_T^2. Any k cover satisfying this condition defines a common subgraph of order k, i.e. all common subgraphs of order k (if any) of two non labeled graphs $G_1(N_1, E_1)$ and $G_2(N_2, E_2)$ are defined by those k-covers of table N that map edges of G_1 onto edges of G_2 and non-edges of G_1 onto non-edges of G_2.

Let

$$(1) \qquad (n_i^1, n_j^1) \nsim (n_k^2, n_l^2)$$

state that pair of nodes n_i^1, n_j^1 (edge or non-edge) of G_1 *is not* mapped on pair of nodes n_k^2, n_l^2 of G_2.

The following proposition can now be stated.

PROPOSITION 1. Two non-labeled graphs $G_1(N_1, E_1)$ and $G_2(N_2, E_2)$ have a common subgraph of order k if and only if there exists a k-cover of their Node Correspondence Table, satisfying the lists of contraints:

(2) $(n_i^1 n_j^1) \nsim (n_k^2, n_l^2)$, for all i, j, such that $(n_i^1, n_j^1) \in E_1$, and all k, l such that $(n_k^2, n_l^2) \notin E_2$.

(3) $(n_i^1, n_j^1) \nsim (n_k^2, n_l^2)$, for all i, j, such that $(n_i^1, n_j^1) \notin E_1$, and all k, l, such that $(n_k^2, n_l^2) \in E_2$.

It was shown elsewhere [7, 8] that the above considered constraints generate incompatibilities between cells of the Node Correspondence Table. The constraint $(n_i^1, n_j^1) \nsim (n_k^2, n_l^2)$ states that n_j^1 cannot be mapped onto n_l^2 if n_i^1 is mapped onto n_k^2 (this is true only for directed graphs). Two cells (i, k) and (j, l) are incompatible, i. e. they cannot belong to the same k cover. The pair of cells $((i, k), (j, l))$ is called *incompatible pair*. If graphs G_1 and G_2 are undirected, the constraint $(n_i^1, n_j^1) \nsim (n_k^2, n_l^2)$ generates two incompatible pairs, namely $((i, k), (j, l))$, and $(i, l), (j, k)$. The *list of incompatible pairs* is the set of all the incompatible pairs derived from constraints (2) and (3). A k-cover S is a *compatible k-cover,* if no pair of cells in S belongs to the list of incompatible pairs. Proposition 1 can now be stated in a different form.

PROPOSITION 2. Two non-labeled graphs $G_1(N_1, E_1)$ and $G_2(N_2, E_2)$ have a common subgraph of order k if and only if there exists a compatible k-cover of their Node Correspondence Table.

An algorithm for deriving compatible k-covers (common subgraphs of order k) can be obtained by a suitable generalization of the procedure described in [7], which is a partially enumerative procedure based on cell selections and table reductions. A different approach is suggested here. Let us consider a binary square table C, called *Compatibility Table*, having the non-zero cells of the Node Correspondence Table on its rows and its columns. $C(i, j)$ has the value 1 if cells i and j are compatible, 0 otherwise. The table is obviously symmetric. The problem of deriving compatible k-covers can be very easily represented on the Compatibility Table. A first set of incompatibilities is immediately derived from the list of incompatible pairs. In addition, a cell is incompatible with any other cell lying on the same row or the same column, due to the definition of k-cover (*k-cover incompatible pairs*). Let α be the set of non-zero cells of the Node Correspondence Table. A *compatibility k-class* γ is a subset (of cardinality k) of α, such that all the elements of γ are pairwise compatible. The existence of a compatible k-cover of the Node Correspondence Table is equivalent to the existence of a compatibility k-class on the Compatibility Table. Therefore, the following proposition is true.

PROPOSITION 3. Two non-labeled graphs $G_1(N_1, E_1)$ and $G_2(N_2, E_2)$ have a common subgraph of order k if and only if there exists a compatibility k-class on their Compatibility Table.

Table C can be considered the adjacency matrix of an undirected graph Γ. A compatible k-class is a complete subgraph of order k of Γ (3).

PROPOSITION 4. Two non-labeled graphs $G_1(N_1, E_1)$ and $G_2(N_2, E_2)$ have a common subgraph of order k if and only if graph Γ has a complete subgraph of order k.

Therefore, procedures for deriving maximal common subgraphs (and for testing subgraph isomorphism) can be based on the derivation of compatibility classes or complete subgraphs, for which several algorithms exist [**9, 10, 11**].

3. Compatibility Table reduction.

When no zero cells exist in the Node Correspondence Table, the Compatibility Table has $m \times n$ rows and columns: its size can be very large, hence the « compatibility classes » algorithm computation time is bound to become substantial. However, Compatibility Table size c.n be reduced, by creating zero cells in the Node correspondence Table, if graphs G_1 and G_2 are labeled, if they have loops or if the only subgraph isomorphism problem is considered. In the following, rules will be given for table reduction: such rules are based on necessary conditions for node correspondence which can be very easily verified.

Rule 1. (Labeled graphs). A zero cell is created in table N for a pair of nodes n_i^1, n_j^2 if n_i^1 and n_j^2 have different labels.

Rule 2. (Graphs with loops). A zero cell is created in table N for a pair of nodes n_i^1, n_j^2, if $(n_i^1, n_i^1) \in E_1$ and $(n_j^2, n_j^2) \notin E_2$ or $(n_i^1, n_i^1) \notin E_1$ and $(n_j^2, n_j^2) \in E_2$.

If we confine ourselves to the subgraph isomorphism problem, the following rules 3 and 4 can be applied to (possibly) reduce the Compatibility Table size. These rules are based on comparisons of *degrees* of nodes of G_1 and G_2 (4).

(3) A graph $G(N, E)$ is *complete* if, for any pair of elements x, y belonging to N, $(x, y) \in E$.

(4) Let G be an unlabeled directed graph. The *inward semi-degree* of a node x is the number of edges leading to x, while the *outward semi-degree* is the number of edges leaving x. If G is undirected the above semi-degrees have the same value (*degree*). If graph G is labeled, degree and semi-degrees can be defined as sets of edge labels.

Rule 3. (non-labeled graphs) A zero cell is created in table N for a pair of nodes n_i^1, n_j^2, if the degree of n_j^2 is less than the degree of n_i^1.

Rule 4 (labeled graphs). A zero cell is created in table N for a pair of nodes n_i^1, n_j^2, if the degree of n_i^1 is not a subset of the degree of n_j^1.

Conditions in rules 3 and 4 need only be satisfied by one of the semi-degrees when applied to directed graphs.

Propositions in Section 2 hold for non-labeled graphs only. If labeled graphs are considered, a different list of constraints appears in proposition 1.

(4) $(n_i^1, n_j^1) \not\sim (n_k^2, n_l^2)$, for all i, j, k and l such that edges (n_i^1, n_j^1) and (n_k^2, n_l^2) have different labels (see definition of isomorphism for labeled graphs). If the list of incompatible pairs is derived from constraints (4), propositions 2,3 and 4 are valid for labeled graphs also.

4. The Algorithm.

Common subgraphs of graphs G_1 and G_2 can be derived in a 2-step procedure:

i) Construct the Compatibility Table C.

ii) Apply to C a procedure for deriving compatibility classes (or complete subgraphs).

In order to obtain the Compatibility Table, we don't need to go through the Node Correspondence Table construction. Candidate node correspondences, according to rules 1, 2, 3 and 4 are stored in a list L, which associates pairs of corresponding nodes to Compatibility Table rows and columns. Zero entries could be derived from the list of incompatible pairs and from the k-cover incompatible pairs. However, an incompatible pair can relate to a pair of corresponding nodes which actually does not exist in L. In fact, L does not contain, in general, all the possible node correspondences, due to the application of rules 1 to 4. As an example, assume our problem is testing subgraph isomorphism for graphs in Fig 2. According to rule 3,

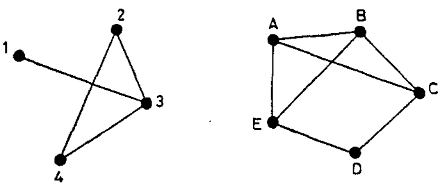


Fig. 2
A pair of graphs tested for subgraph isomorphism; node 3 cannot be mapped onto node D.

node 3 cannot correspond to node D, so that node correspondence $(3, D)$ does not appear in L. On the other hand, the list of incompatible pairs contains the incompatible pair $((1, A), (3, D))$, generated from constraint $(1,3) \nsim (A, D)$. The incompatibility given by $((1, A), (3, D))$ cannot be represented on the Compatibility Table, because no row is associated to cell $(3, D)$. As a matter of fact, the incompatible pair $((1, A), (3, D))$ can be disregarded. Actually, zero entries in the Compatibility Table are created by testing for compatibility every pair against all other pairs in L. Let $(a, b) \underline{\nsim} (c, d)$ denote that pairs of nodes (a, b) and (c, d) (edges or non edges of G_1 and G_2 respectively) have different labels or that only one of them is an edge. Pair $((i, j), (k, l))$ is incompatible (a zero is entered in the Compatibility Table) if one of the following conditions is verified.

a) (directed graphs)

i) $(i, k) \underline{\nsim} (j, l)$ or $(k, i) \underline{\nsim} (l, j)$ (List of incompatible pairs), or

ii) $i = j$ or $k = l$ (k-cover incompatible pairs)

b) (undirected graphs)

i) $(i, k) \underline{\nsim} (j, l)$ List of incompatible pairs), or

ii) $i = j$ or $i = l$ or $k = j$ or $k = l$ (k-cover incompatible pairs)

The algorithm in [10] was chosen for the derivation of compatibility classes, because of its simplicity and its efficiency for medium-size compatibility tables. This algoritm derives compatibility classes by a tree-branching technique. A compatibility class of cardinality k corresponds to a path of lenght k on the tree. The basic operation is:

i) Search for a « 1 » in a column of the table.

ii) Execute the logical AND of corresponding elements of two table columns (5).

Graph G_1, of order m, is isomorphic to a subgraph of G_2, if there exists a maximal compatibility m-class (complete subgraph of Γ of order m). Therefore the algorithm searches for compatibility classes whose cardinality is exactly m. Compatibility classes corresponding to paths whose lenght is less than m can be disregarded. A parameter, the maximum expected lenght, is associated to every new branch. When its value is less than m, the branch is on a « dead-end path » and the branching process along that path is stopped.

---

(5) Note that it can be convenient to store binary table C in a packed form, because of its size. AND-ing of corresponding elements can be efficiently executed on packed columns.

As far as the derivation of maximal common subgraphs is concerned, let us first note that an upper bound for the cardinality of compatibility classes is given by the least (say m) of the orders of G_1 and G_2.

Therefore branching is carried on down to the m-th level, in the worst case.

All maximal common subgraphs correspond to maximal compatibility classes while the converse is not true. Assume, for example, the following maximal compatibility classes have been derived:

$$A: \{(1,2), (2,4), (3,5)\}$$

$$B: \{(1,4), (4,2), (3,3), (2,5)\}.$$

Class A is maximal, since it is not contained in class B. However, sets of nodes (subgraphs) $\{1, 2, 3\}$ and $\{2, 4, 5\}$ defined by A are both subsets of sets of nodes $\{1, 2, 3, 4\}$ and $\{2, 3, 4, 5\}$ defined by B. Because of our definition of maximal common subgraphs, maximal compatibility class A does not define a maximal common subgraph. Therefore common subgraphs derived from maximal compatibility classes, must be tested for maximality.

In several applications it is also required that maximal common subgraphs be connected. In this case, subgraphs are splitted into connected components. Such components are then tested for maximality. Let us consider labeled directed graphs G_1 and G_2 in Fig. 3. Their adjacency matrices are shown in Fig. 4 and their compatibility table is shown in

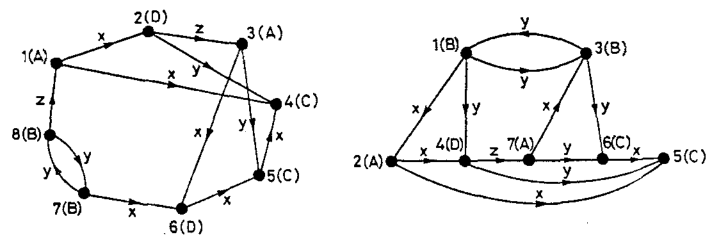


Fig. 3. Sample labeled directed graphs G_1 and G_2.

	1	2	3	4	5	6	7	8
1	A	x		x				
2		D	z	y				
3			A		y	x		
4				C				
5				x	C			
6					x	D		
7						x	B	y
8	z						y	B

	1	2	3	4	5	6	7
1	B	x	y	y			
2		A		x	x		
3	y		B			y	
4				D	y		z
5					C		
6					x	C	
7			x			y	A

Fig. 4. Adjacency matrices of graphs G_1 and G_2 in Fig. 3.

Fig. 5. Table 1 shows the maximal compatibility classes. Maximal (connected)

	(1,2)	(1,7)	(2,4)	(3,2)	(3,7)	(4,5)	(4,6)	(5,5)	(5,6)	(6,4)	(7,1)	(7,3)	(8,1)
(1,7)	0												
(2,4)	1	0											
(3,2)	0	1	0										
(3,7)	1	0	1	0									
(4,5)	1	0	1	0	1								
(4,6)	0	0	0	1	0	0							
(5,5)	0	1	0	0	0	0	0						
(5,6)	1	0	1	0	1	1	0	0					
(6,4)	0	0	0	1	0	0	1	0	0				
(7,1)	0	1	0	0	1	1	1	1	1	0			
(7,3)	1	0	1	1	0	1	0	1	0	0	0		
(8,1)	0	0	0	0	1	1	1	1	1	0	0	1	
(8,3)	0	0	1	1	0	1	0	1	0	1	1	0	0

Fig. 5. Compatibility Table of graphs G_1 and G_2 in Fig. 3.

common subgraphs are shown in Table 2. Maximal common subgraph *I* corresponds to maximal compatibility class 1. No other class generates connected common subgraphs. Maximal common subgraphs II, III and IV are obtained from maximal compatibility classes 7 (or 8), 9 (or 11) and 10 (or 12) respectively. They are the only maximal connected components found in our example. In Fig. 6 graphs G_1 and G_2 are displayed and maximal

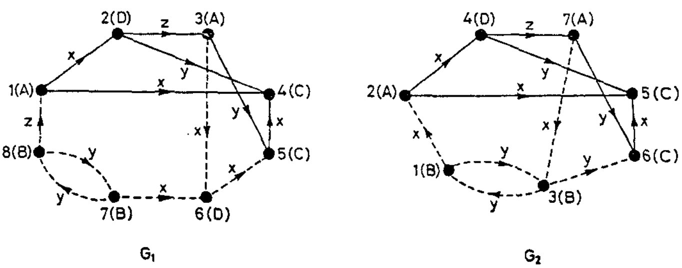


Fig. 6. Maximal common subgraph *I* of graphs G_1 and G_2.

common subgraph *I* is exhibited.

The first step of the procedure described in this note (compatibility Table construction) is quite simple. The computation time depends, in the worst case, on $m^2 \times n^2$, where m and n are the orders of G_1 and G_2. It is rather hard to define a meaningful worst case for the second step (maximal compatibility classes derivation). The computation time is strongly dependent on table size and on number of 1's in the table. When applied to labeled graphs the computation time is reasonable even with large graphs.

A FORTRAN IV program implementing the above algorithm has been written and run on an IBM 360/67. The program is available at the Istituto di Elaborazione della Informazione, Pisa.

Table 1. Maximal compatibility classe derived from Compatibility Table in Fig. 5.

1.	{(1,2), (2,4), (3,7), (4,5), (5,6)}
2.	{(1,2), (2,4), (4,5), (7,3)}
3.	{(3,7), (4,5), (5,6), (7,1)}
4.	{(3,7), (4,5), (5,6), (8,1)}
5.	{(1,7), (5,5), (7,1)}
6.	{(2,4), (4,5), (8,3)}
7.	{(3,2), (4,6), (6,4)}
8.	{(3,2), (6,4), (8,3)}
9.	{(4,5), (7,1), (8,3)}
10.	{(4,5), (7,3), (8,1)}
11.	{(5,5), (7,1), (8,3)}
12.	{(5,5), (7,3), (8,1)}
13.	{(1,7), (3,2)}
14.	{(3,2), (7,3)}
15.	{(4,6), (7,1)}
16.	{(4,6), (8,1)}

Table 2. Maximal (connected) common subgraphs of graphs G_1 and G_2 in Fig. 3.

I.	{1, 2, 3, 4, 5}	{2, 4, 7, 5, 6}
II.	{3,6}	{2,4}
III.	{7,8}	{1,3}
IV.	{7,8}	{3,1}

REFERENCES

[1] UNGER, S. H., *GIT, a heuristic program for testing pairs of directed line graphs for isomorphism,* Comm. ACM 7 (1964), 26-34.

[2] SUSSENGUTH, E. H. JR., *A graph-theoretical algorithm for matching chemical structures,* J. Chem. Doc. 5 (1965), 36-43.

[3] SALTON, G. and SUSSENGUTH, E. H. JR., *Some flexible information retrieval systems using structure matching procedures,* Proc. AFIPS 1964 SJCC 25 (1964), Spartan Books, New York, 587-598.

[4] CORNEIL, D. G. and GOTLIEB, C.C., *An efficient algorithm for graph isomorphism,* J. ACM 17 (1970), 51-64.

[5] MORPURGO, R., *Un metodo euristico per la verifica dell'isomorfismo di due grafi non orientati,* Calcolo 8 (1971), 1-31.

[6] SIROVICH, F., *Isomorfismo fra grafi : un algoritmo efficiente per trovare tutti gli isomorfismi,* Calcolo 8 (1971), 301-337.

[7] LEVI, G. and LUCCIO, F., *A technique for graph embedding with constraints on node and arc correspondences,* Information Sciences 5 (1973), 1-23.

[8] LEVI, G. and LUCCIO, F., *A weighted graph embedding technique and its application to automatic circuit layout,* Calcolo 8 (1971), 49-60.

[9] PAULL, M. C. and UNGER, S. H., *Minimizing the number of states in incompletely specified sequential switching functions,* IRE Trans. on Electronic Computers 8 (1959), 356-367.

[10] GRASSELLI, A., *A note on the derivation of maximal compatibility classes,* Calcolo 3 (1966), 165-176.

[11] WOLFBERG., M. S., *Determination of maximally complete subgraphs,* Moore School of E E., Rept. n. 65-27, Univ. of Pennsylvania, Philadelphia, Penn. (1965).