match no. 28 pp. 29-43 1992

A HEURISTICAL POLYNOMIAL ALGORITHM FOR ORBITS OF NONDIRECTED, CONNECTED FINITE DIGRAPHS WITH AT LEAST TWO VERTICES

G. BIESS, M. BAUTZ
Department of Mathematics and Informatics, Technische Hochschule "Carl Schorlemmer" Leuna-Merseburg, O-4200 Merseburg

(received: May 1990)

Summary: A polynomial algorithm is presented which in many cases yields the orbits of an undirected, connected, finite digraph. The algorithm has been programmed in Turbo Pascal and checked with many examples.

1. INTRODUCTION

An automorphism of a graph $G = [V,E]$ is defined as a bijective mapping $\sigma : V \to V$, a permutation of the vertices, which always transmits edges into edges i.e. which transforms G into itself. The set of all the automorphisms of G forms a group [1]. Related to this group is a partition of V into classes, called the orbits of G. Two vertices $x, y \in V$ belong to the same orbit if and only if an automorphism exists, which transmits x into y. Therefore an orbit exactly includes all those vertices which have the same properties relative to G. In this sense the orbits can be regarded as the "finest classification" of V.

It was shown in [2] that the following two decision problems are polynomially equivalent:

(I) Given a graph G and two of its vertices x,y,
Decide: Are x,y numbers of the same orbit of G ?

(II) Given two graphs $G = [V,E]$ and $G' = [V',E']$, each of which has n vertices and m edges. Decide: Is there an isomorphism $\sigma : V \to V'$ which transforms G into G' ?

Knowing the orbits we can easily determine automorphisms of the graph. If the orbits of two graphs G, G' are known it

is not difficult to decide, whether or not they are isomorphic, and isomorphisms can be determined explicitely if desired.
Such problems also occur in chemistry, for instance when isomers are studied, or if one has to examine whether or not a given structure is already contained in a database.

Due to this equivalence of problems, algorithms are wanted which are capable of determining the orbits of a given graph. The present paper is intended to make a contribution to this problem. An algorithm is presented (divided into four subalgorithms) which is polynomial in time. It was applied many times to a comprehensive class of graphs (characterized in the following), and in all cases the orbits were found. But up to this time no complete proof could be given. Therefore the algorithm still must be called heuristical.
In the following the four subalgorithms are described. Figure 1 shows how the individual parts co-operate.

2. Subalgorithm TA 1

We assume that graph $G = [V,E]$ is given, where $V = 1_n = \{1,2,\ldots,n\}$ is the set of vertices and E the set of edges. Let $M_{i\lambda}$ be the set of all vertices, the distance of which to vertex i is λ, $M_{i0} = \{i\}$. We will call $M_{i\lambda}$ the "level λ of vertex i". If l_i is the highest level, then $M_{i\lambda} \neq \emptyset$ for $0 \leq \lambda \leq l_i$, and $M_{i\lambda} = \emptyset$ for $\lambda > l_i$. Let $m_{i\lambda}$ be the cardinality of $M_{i\lambda}$.
Now we form the sequences

$$\mathfrak{M}_i := (M_{i\lambda}) \quad , \mathfrak{w}_i^0 := (m_{i\lambda}) \qquad (1)$$
$$\lambda = 0(1)l_i \quad , \quad i \in 1_n .$$

Two vertices $i,j \in 1_n$ are called "equivalent of 1st stage" if and only if $\mathfrak{w}_i^0 = \mathfrak{w}_j^0$. By this relation a classification $\mathfrak{K}_1$ of V is produced, the classes of which may be arbitrarily numbered:

$$\mathfrak{K}_1 = \{K_{11}, K_{12},\ldots,K_{1k_1}\} \quad . \tag{2}$$

Obviously, an algorithm TA I, performing these operations is polynomial in time.

3. Subalgorithm TA II

Now the case of $\mathfrak{K}_1$ having more than one class is considered ($k_1 \geq 2$). Starting from $\mathfrak{K}_1$, we sucessively produce a sequence

$$\mathfrak{F} := (\mathfrak{K}_r) \quad , \quad r = 1(1)r_o \tag{3}$$

where for $1 \leq r \leq r_o-1$ classification $\mathfrak{K}_{r+1}$ is finer than $\mathfrak{K}_r$. The sequence obviously ends after at most n-2 steps ($r_o \leq n-1$).

The equivalence relation corresponding to the classification $\mathfrak{K}_r$ is called "equivalence of stage r". The sequence $\mathfrak{F}$ is generated by subalgorithm TA II, which works as follows. Assume that classification $\mathfrak{K}_r$ has already been produced. The classes of $\mathfrak{K}_r$ may be arbitrarily numbered from 1 to k_r, $\mathfrak{K}_r = \{K_{r1}, K_{r2},\ldots,K_{rk_r}\}$. Now each vertex j is assigned a label q_j^r:

$$q_j^r = \varkappa \iff j \in K_{r\varkappa} \quad . \tag{4}$$

After that, in each $M_{i\lambda}$ the vertices j are substituted by the labels q_j^r, thereby transforming the sets $M_{i\lambda}$ into families $F_{i\lambda}^r$. We then have for each vertex $i \in I_n$ a sequence

$$\mathfrak{m}_i^r := (F_{i\lambda}^r) \quad , \quad \lambda = 0(1)l_i \tag{5}$$

of l_i+1 families instead of the former sequence $\mathfrak{M}_i = (M_{i\lambda})$. Two vertices i,j are said to be "equivalent of stage r+1"

if and only if $w_i^r = w_j^r$; i and j will then belong to the same class within the classification $\mathfrak{K}_{r+1}$ (this is also valid for r = 0). As long as $\mathfrak{K}_{r+1}$ is finer than $\mathfrak{K}_r$ (i.e., at least one class $K_{r\varkappa}$ of $\mathfrak{K}_r$ is divided into at least two classes of $\mathfrak{K}_{r+1}$) the procedure is repeated. Finally we get a classification

$$\mathfrak{K}^* := \mathfrak{K}_{r_o} = \{K_{r_{o1}}, K_{r_{o2}}, \ldots, K_{r_o k_o}\} \tag{6}$$

with at least two classes.

After TA II is finished the second part of subalgorithm TA III follows, see also figure 1. The two sections of TA III will now be described.

4. Subalgorithm TA III

We consider the case of classification $\mathfrak{K}_1$, generated by TA I, having only a single class. As can be seen from figure 1, TA II is not applied in this case, because it would yield nothing but this single class again.

$\mathfrak{K}_1 = \{K_{11}\}$ means that all vertices $i \in I_n$ are equivalent of 1st stage, i.e. $w_i^o = w_j^o$, $\forall i, j \in I_n$, see (1).

Therefore, in the case under consideration, each sequence $\mathfrak{M}_i$ has the same number λ of levels, and

$$|M_{i\lambda}| = |M_{2\lambda}| = \ldots = |M_{n\lambda}| , \quad = O(1)\lambda. \tag{7}$$

Such a graph is called "regular in distance".
The sequence of levels $\mathfrak{M}_i = (M_{io}, \ldots, M_{i\lambda})$ can be regarded as a classification of V due to the relation "equal distance to vertex i". For a graph which is regular in distance we therefore have a system of n classifications

$$S_I = \{\mathfrak{M}_1, \mathfrak{M}_2, \ldots, \mathfrak{M}_n\} \tag{8}$$

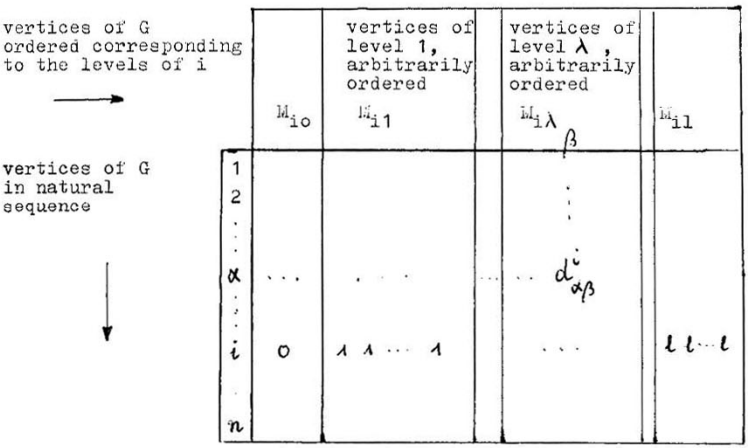
as a starting point for a subalgorithm TA III which could be able to refine $\mathfrak{K}_1$. TA III is divided into two sections the

first of which will now be described.

4.1 First part TA 1II1

Starting from S_I for each vertex $i \in I_n$ a specific scheme D^i, and from this a conjugated classification $\mathcal{Z}_i$ is generated. Matrix $D^i = (d^i_{\alpha\beta})$ corresponding to vertex i is constructed as shown in figure 2:

FIGURE 2: Structure of matrix D^i



Element $d^i_{\alpha\beta}$ lies in the crossing point of row α and the column of vertex ß. $d^i_{\alpha\beta} = s$, if vertex α is a number of level $M_{\beta s}$ (and vertex ß a number of $M_{\alpha s}$). Obviously, $d^i_{\alpha\alpha} = 0$ and $d^i_{\alpha\beta} = d^i_{\beta\alpha}$. Matrix D^i can be regarded as a modified adjacency matrix, the latter being represented by the 1-elements in D^i. Finally we assign the sequence $(M_{io}, M_{i1}, \ldots, M_{il})$ as the "top row" to matrix D^i. Within the sets $M_{i\lambda}$ the vertices can be ordered arbitrarily. For graphs regular in distance the following property is characteristic. Let λ be an arbitrary element of $I_l = \{0,1,\ldots,l\}$. Then, in each row and in each column of D^i, $i \in I_n$, λ occurs $m_{i\lambda}$ times.

Because the same column stands under the same vertex ß of the top row in each D^i it is sufficient to store D^1 and all the top rows only. From this data all matrices D^i can be obtained by permuting the columns of D^1. The column under vertex ß in the top row denotes in which levels of ß all the other vertices are present. Similarly, row k of matrix D^i describes the levels of k in which the vertices of the top row exist.

Using D^i a partition $\mathfrak{Z}_i$ of V is formed and assigned as a classification to vertex i. To this end we use the criterion "equality of rows". Two rows j,k of D^i are said to be equal if and only if under each segment $M_{i\lambda}$ of the top row the rows j and k coincide in the numbers of 0-, 1-,...,l-elements. We then say that in D^i the vertices j,k have "equal rows" or "equal row pattern".

The resulting classification $\mathfrak{Z}_i$ correlates with a certain vertex i, i.e., to each vertex $i \in I_n$ a partition of V is assigned. In such a case it is called a "1st kind classification" (1.k.c.). On the other hand a partition of V, which does not correspond to a certain vertex, is called a "2nd kind classification" (2.k.c.). For example in TA I the $\mathfrak{M}_i$ are 1.k.c., whereas $\mathfrak{K}_1$ is a 2.k.c. .

Returning to TA III1, $\mathfrak{Z}_i$ is in general finer than $\mathfrak{M}_i$ produced by TA I. We then have a new system of 1.k.c.

$$S_{II} = \{\mathfrak{Z}_1, \mathfrak{Z}_2, \ldots, \mathfrak{Z}_n\} \qquad (9)$$

In TA I the pertinent 2.k.c. was $\mathfrak{K}_1$ (which in the case under consideration has a single class only).

TA III1 also leads to a 2.k.c. using the criterion "equality in partition". Two partitions $\mathfrak{Z}_i$, $\mathfrak{Z}_j$ are said to be equal if and only if there is a bijective mapping of the classes of $\mathfrak{Z}_i$ to those of $\mathfrak{Z}_j$, so that every two corresponding classes are equal in cardinality and in row pattern. The 2.k.c. obtained in this way is denoted by $\mathfrak{K}^*$.

Concerning the numbering of the classes in S_{II} we now

establish some rules (see table 1):

TABLE 1: Systems S_I, S_{II} of f.k.c. ($r_i \geq 1$, $\forall i \in I_n$)

vertex i	$\mathfrak{M}_i$	$\mathfrak{Z}_i$	
1	$M_{10}\ M_{11} \cdots M_{1l}$	$Z_{10}\ Z_{11} \cdots Z_{1r_1}$	S_{II}
2	$M_{20}\ M_{21} \cdots M_{2l}$	$Z_{20}\ Z_{21} \cdots Z_{2r_2}$	
⋮			
i	$M_{io}\ M_{i1} \cdots M_{il}$	$Z_{io}\ Z_{i1} \cdots Z_{ir_i}$	
⋮			
n	$M_{no}\ M_{n1} \cdots M_{nl}$	$Z_{no}\ Z_{n1} \cdots Z_{nr_n}$	
	S_I		

In all cases $Z_{io} = \{i\}$. The other classes $Z_{i1},\ldots,Z_{ir_i}$ can be numbered arbitrarily, but ensuring that, if $\mathfrak{Z}_i$, $\mathfrak{Z}_j$ are equal in partition, every two corresponding classes have the same second index. Then the following is true. If two classes correspond to one another, then they have the same second index, but if the second indices are equal, the classes do not necessarily correspond. If $r_i=1$, $\forall i \in I_n$, we call S_{II} a reproduction of S_I. Then $\mathfrak{K}^* = \mathfrak{K}_1$, i.e., the new s.k.c. also has only one class, no refinement has resulted from TA III1. Otherwise S_{II} is finer than S_I, and as a consequence $\mathfrak{K}^*$ has at least two classes. Note that each new class obtained from TA III1 is entirely contained in one of the old classes of S_I, i.e. the old classes always are junctions of new classes.

As just mentioned the s.k.c. $\mathfrak{K}^*$ resulting from TA III1 either again has only one class or gives a refinement in partitioning V. In the first case, the subalgorithms described in the following also are not able to find any finer classification, the analysis of the graph cannot be completed with our algorithms. In the other case, where $\mathfrak{K}^*$ has at least two classes (resulting from TA III1 or from TA II), a second part of TA III is applied. As can be seen in figure 1, the

branches TA I - TA II and TA I - TA III1 of the whole algorithm converge at this point.

4.2 Second part TA III2

To simplify matters TA III2 is described here by means of an example. First we number the r classes of $\mathfrak{K}^*$ arbitrarily by natural numbers:

$$\mathfrak{K}^* = \{K_1^*, K_2^*, \ldots, K_r^*\} \qquad (10)$$

It is obvious that in a (possibly existing) f.k.c. $\mathfrak{Z}_i'$ which is finer than $\mathfrak{Z}_i$, only those vertices can form a class, which lie in the same class also in $\mathfrak{K}^*$. Therefore a finer partition can result from $\mathfrak{Z}_i$ by cutting each class Z_{ij} in turn with all the classes of $\mathfrak{K}^*$. Let us look at an example, where G has 15 vertices. Assume that from TA III1

$$\mathfrak{Z}_1 = \{Z_{10}, \ldots, Z_{13}\} = \{\{1\}, \{2,3,4,5,6,7\}, \{8,9\}, \{10,11,12,13,14,15\}\}$$

was obtained, and that the corresponding s.k.c. is

$$\mathfrak{K}^* = \{K_1^*, \ldots, K_5^*\} = \{\{1,5,8\}, \{2,3,9,10,11,12\}, \{13\}, \{14,15\}, \{4,6,7\}\}.$$

The cutting procedure then yields the following finer partition from $\mathfrak{Z}_1$:

$$\mathfrak{Z}_1' = \{\{1\}, \{5\}, \{2,3\}, \{4,6,7\}, \{8\}, \{9\}, \{10,11,12\}, \{13\}, \{14,15\}\}.$$

Note that the sequence of classes in $\mathfrak{Z}_1'$ is uniquely fixed by the numbering within $\mathfrak{Z}_1$, $\mathfrak{K}^*$ and the cutting procedure described above.

Performing the same procedure with all $\mathfrak{Z}_i$, i=1(1)n, leads to a new system S_{II}' of f.k.c. . S_{II}' is either equal to S_{II} ("reproduction of S_{II}") or can be finer than S_{II}.

In the last case, applying the criterion "equality in partition" to S'_{II}, a new s.k.c. is obtained. Taking

$$\mathfrak{K}^* := \mathfrak{K}^{*\prime}\ ,\quad \mathfrak{Z}_i := \mathfrak{Z}'_i\ ,\quad i=1(1)n \qquad (11)$$

the procedure described above is repeated until $S'_{II} = S_{II}$. Then TA III2 is finished. The final results may again be denoted as S_{II}, $\mathfrak{Z}_i$, $\mathfrak{K}^*$.

Note that, if the branch TA I - TA II - TA III2 is used, the classifications $\mathfrak{M}_i$ take the place of $\mathfrak{Z}_i$.

After TA III2 is finished the procedure is continued by subalgorithm TA IV, described in the following section.

5. Subalgorithm TA IV

One can see from the following description, that after all subalgorithm TA IV is nothing but a modification of TA III. It starts with the system of f.k.c. S_{II} and the s.k.c. $\mathfrak{K}^*$, obtained by TA III2. Now the first operation is to number the classes of S_{II}, i.e. of the f.k.c. $\mathfrak{Z}_p$ as follow: In $\mathfrak{Z}_p$ the class consisting of vertex p itself is always denoted by number 1. All other classes are numbered arbitrarily in succession, but the following rule has to be observed: If the classifications $\mathfrak{Z}_p$, $\mathfrak{Z}_q$ are equal in partition, every two corresponding classes must have the same number. After numbering the system shall be denoted as $\bar{S}_{II}$. Now for each vertex $i \in I_n$ a "label column vector, (l.c.v.)" $w^{(i)}$ is formed,

$$w^{(i)T} = (w_1^{(i)} \ \dots\ w_n^{(i)}),\quad i=1(1)n\ ,$$

where $w_k^{(i)}$ is the number of that class in $\mathfrak{Z}_k$ to which vertex i belongs. Therefore the components of vector $w^{(i)}$ state the classes of the different classifications $\mathfrak{Z}_k$, $k=1(1)n$ in which vertex i lies. With the help of these l.c.v. we now form a "label" scheme, (l.s.) Δ^i for each vertex i. $\Delta^i = (\delta^i_{\kappa j})$ is a matrix (analogous to D^i in

TA III) with n+1 rows and v(i) columns, where v(i) is the number of neighbours of vertex i in G. Δ^i is arranged as follows: The top row (k=0) contains all the neighbours of vertex i (in an arbitrary order), and under each neighbour j the pertinent l.c.v. $w^{(j)}$ is located. Therefore it is sufficient to store all the l.c.v., the sets of neighbours are already given by the levels M_{i1}, obtained in TA I.

As already mentioned, Δ^i takes the place of D^i in TA III1. Therefore in an analogous manner the criterion "equality of rows" can be applied and yields a f.k.c. $\mathfrak{Z}'_i$ for each vertex i, which is either equal to $\mathfrak{Z}_i$ in the old system S_{II} or is a refinement of $\mathfrak{Z}_i$. As long as a refinement occurs for any $\mathfrak{Z}_i$ the procedure is repeated.

Let be $\bar{S}_{r_o}$ the system finally obtained.

An s.k.c. $\bar{\mathfrak{K}}$ is generated, as in TA III1, using the criterion "equality of partition".

Concerning the form of $\bar{S}_{r_o}$ and $\bar{\mathfrak{K}}$, the following cases are possible:

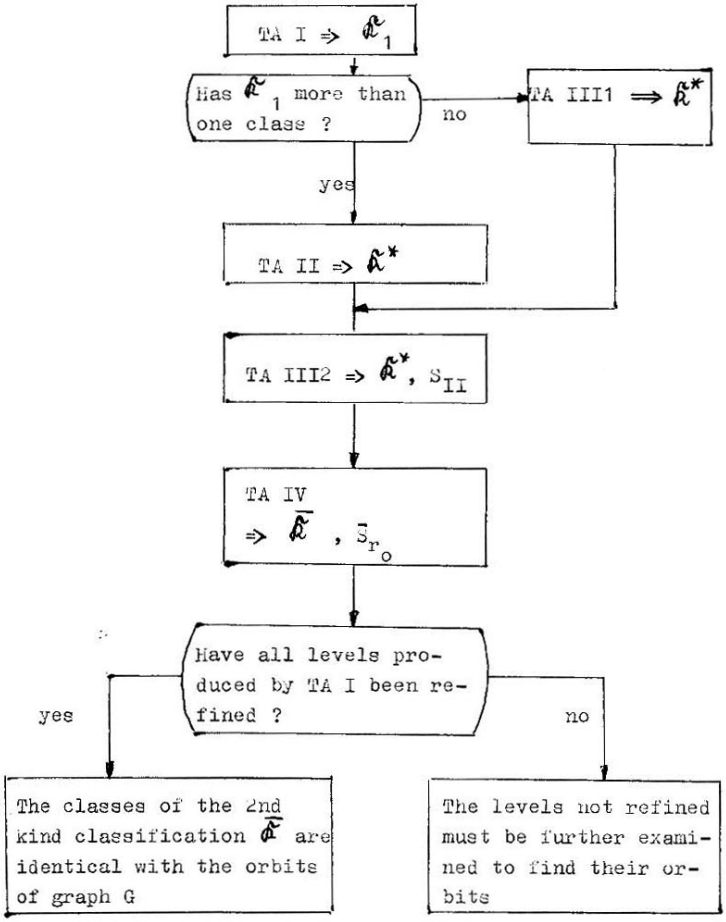
(i) $\bar{S}_{r_o}$ consists of n different f.k.c., and therefore has n classes. Then each single vertex of G also is an orbit of G

(ii) $\bar{S}_{r_o}$ consists of n equal f.k.c., and therefore $\bar{\mathfrak{K}}$ has a single class. Then G has only one orbit, viz. the complete set V of vertices.

(iii) $\bar{S}_{r_o}$ consists of at least two and at most n-1 different f.k.c. . Then every class of $\bar{\mathfrak{K}}$ is an orbit of G.

As already mentioned at the beginning a complete proof cannot be given at this time. Many examples have been treated. The algorithm was successful in all cases without any exception, when all levels produced by TA I were divided into at least two classes by applying TA II - TA IV.

Figure 1: Flow sheet of the complete algorithm



Complexity of the algorithm

Finally a rough estimation concerning the running time of the algorithm will be made (the demand for memory is not considered).

TA I: The following operations must be performed:

1) Establishing the system S_I of r.k.c. $\mathfrak{M}_i$, i=1(1)n: Vertex i itself gets number 0, its neighbours number 1, the neighbours of these number 2 and so on. Using a list of edges $O(n^2)$ steps are sufficient to find the levels relative to vertex i. Since there are n vertices, $O(n^3)$ is an upper bound for determining all levels of S_I.

2) Calculation of the sequences $\mathfrak{w}_i^o$, i=1(1)n: In each class of S_I the members have to be counted. The expense is $O(n^2)$.

3) Determination of the s.k.c. $\mathfrak{K}_1$: To find the classes of $\mathfrak{K}_1$, we have to compare $\frac{n}{2}(n-1)$ vectors m_i^o. Since n is the maximum number of components, this step of TA I is limited by $O(n^3)$.

4) Numbering the classes within $\mathfrak{K}_1$ is a process with O(n). Summarising the demand of TA I yields $O(n^3)$.

TA II: We suppose that by TA I the levels of S_I are stored in the form of a list (containing n^2 natural numbers):

$$\{1\}\,|m_1^{(1)}|\ldots|m_{k_1}^{(1)}|\;\{2\}|m_1^{(2)}|\ldots|m_{k_2}^{(2)}|\ldots\{n\}|m_1^{(n)}|\ldots|m_{k_n}^{(n)}|.$$

TA II is applicable if TA I yields at least two classes. Then the following operations must be performed:

1) The elements of the classes of S_I have to be substituted by the corresponding class numbers of $\mathfrak{K}_r$, the effort being $O(n^3)$.

2) Comparison of the sequences of families to get the new s.k.c. $\mathcal{A}_{r+1}$. For every two sequences the demand is $O(n^2)$. In the worst case $\frac{n}{2}(n-1)$ such pairs have to be compared. Therefore step 2 is of order $O(n^4)$.
It follows that TA II as a whole is limited by $O(n^4)$.

TA III 1 Necessary arrangements:

1) Forming matrix D^i demands $O(n^2)$ steps, therefore $O(n^3)$ is the order for arranging all matrices.

2) Application of the criterion "equality of rows". $O(n^2)$ comparisons of vectors with n components, i.e. $O(n^3)$ steps, are necessary for each D^i, so $O(n^4)$ results.

3) Determining the s.k.c. $\mathcal{A}^*$ with the help of the criterion "equality of partition" analogous to (2) in TA II, leads to the demand $O(n^4)$.
As a whole we have $O(n^4)$ as an upper bound for TA III 1.

TA III 2 Operations to be done:

1) Cutting the classes of S_{II} with those of $\mathcal{A}^*$ results in a demand of $n \cdot O(n^2) = O(n^3)$.

2) Building up the new s.k.c. $\mathcal{A}^*$, analogous to (3) in TA III 1, leads to $O(n^4)$.

3) It may be necessary to repeat the operations just mentioned, which gives $O(n^5)$ as an upper bound for TA III 2.
It follows that TA III as a whole is limited by $O(n^5)$.

TA IV: Because this subalgorithm essentially is a modified form of TA III, $O(n^5)$ can also be given as an upper bound.

Finally, it can be stated that the whole algorithm (TA I-IV) presented here is of the order $O(n^5)$ and therefore a polynomial one.

7. An example

Finally an example for the application of the algorithm described above will be given:

Graph G is given by its adjacency list, see table 2.

vertex	neighbours	vertex	neighbours	vertex	neighbours
1:	4,7,10,13	11:	4,9,12,17	21:	2,14,20,25
2:	3,7,12,21	12:	2,11,13,28	22:	10,14,23,29
3:	2,4,5,16	13:	1,8,12,26	23:	19,22,24,27
4:	1,3,11,14	14:	4,15,21,22	24:	9,23,25,28
5:	3,6,10,25	15:	14,16,19,27	25:	5,18,21,24
6:	5,7,8,20	16:	3,15,17,20	26:	13,17,18,27
7:	1,2,6,18	17:	11,16,26,29	27:	15,23,26,28
8:	6,9,13,29	18:	7,19,25,26	28:	12,24,27,29
9:	8,10,11,24	19:	15,18,20,23	29:	8,17,22,28
10:	1,5,9,22	20:	6,16,19,21		

Table 2: Adjacency list of graph G

TA 1 yield a s.k.c. $\mathcal{K}_1$, the four classes of which are given in table 3a.

class number i	class K_{1i}	corresponding sequence m^o
1	{1,4,7,10,13,14,17,18 21,22,25,26,29}	(1,4,12,2)
2	{2,3,5,6,8,9,11,12}	(1,4,12,11,1)
3	{15,19,23,27}	(1,4,10,12,2)
4	{16,20,24,28}	(1,4,11,12,1)

Table 3a: Results of TA I

Because $\mathcal{K}_1$ has more than one class, TA II must be applied. This results in a new s.k.c. $\mathcal{K}^*$ with 6 classes, see table 3b.

class number i	members of class i	class number i	members of class i
1	1	5	3,6,9,12
2	4,7,10,13	6	2,5,8,11
3	17,21,25,29	7	15,19,23,27
4	14,18,22,26	8	16,20,24,28

Table 3b: Results of TA II

Application of TA III 2 and TA IV finally yields the orbits of G:

{1} {4,10} {7,13} {3,9} {6,12} {17,25} {21,29} {14,22}

{18,26} {2,8} {5,11} {15,23} {19,27} {16,24} {20,28}

In this example there is only one nonidentical automorphism of G, viz.

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \\ 1 & 8 & 9 & 10 & 11 & 12 & 13 & 2 & 3 & 4 & 5 & 6 & 7 & 22 \end{pmatrix}$$

$$\begin{pmatrix} 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 & 26 & 27 & 28 & 29 \\ 23 & 24 & 25 & 26 & 27 & 28 & 29 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 \end{pmatrix}.$$

References

[1] van der Waerden "Algebra I" Seite 1 Springer-Verlag Berlin-Heidelberg-New York 1971

2 READ, R.C. and D.C. CORNEIL: The graph isomorphism disease. J. Graph Theory 1 (1977), 339 - 363.