COMPUTER GENERATION OF VERTEX-GRAPHS

by

N. S. Sridharan

STAN-CS-73-381
July 1973

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

COMPUTER SCIENCE DEPARTMENT REPORT
NO. STAN-CS-73-381

JULY, 1973

COMPUTER GENERATION OF VERTEX-GRAPHS

by

N. S. Sridharan

ABSTRACT: In connection with the problem of generating all multigraphs having a specified vertex-degree list, Lederberg has presented a tree-generation algorithm which, exploiting a canonical-lexical notational system, constructs the complete set of solutions in canonically increasing order, without redundancy. Brown, et.al., have expanded the scope of generation to cyclic graphs. Their method relies upon the existence of a complete and irredundant set of "vertex-graphs" and the procedure consists of a series of graph labelling steps. Graph labelling allows one to assign labels from a given set to the vertices (edges) of a graph, in such a way that knowing the symmetry group of the graph, equivalent label assignments are avoided prospectively. Recent work has been directed towards supplying the vertex-graphs needed by the generator. A program has been written to generate all graphs with t trivalent and q quadrivalent vertices, from graphs having (t + 2q) trivalent vertices. This method, however, will generate redundant vertex-graphs and some issues in isomorph elimination are considered in this presentation.

This work was supported by ARPA Contract SD-183 and NIH Grant RR00612-03.

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1 ORIGINATING ACTIVITY (Corporate author) Stanford University Department of Computer Science Stanford, California 94305	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3 REPORT TITLE

COMPUTER GENERATION OF VERTEX-GRAPHS

4 DESCRIPTIVE NOTES (Type of report and inclusive dates)

technical report, July 1973

5 AUTHOR(S) (First name, middle initial, last name)

N. S. Sridharan

6 REPORT DATE July 1973	7a. TOTAL NO. OF PAGES 19	7b. NO. OF REFS 9
8a. CONTRACT OR GRANT NO SD - 183 b. PROJECT NO c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) STAN-CS-73-381 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	

10 DISTRIBUTION STATEMENT

Releasable without limitations on dissemination

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY

13 ABSTRACT

In connection with the problem of generating all multigraphs having a specified vertex-degree list, Lederberg has presented a tree-generation algorithm which, exploiting a canonical-lexical notational system, constructs the complete set of solutions in canonically increasing order, without redundancy. Brown, et al., have expanded the scope of generation to cyclic graphs. Their method relies upon the existence of a complete and irredundant set of "vertex-graphs" and the procedure consists of a series of graph labelling steps. Graph labelling allows one to assign labels from a given set to the vertices (edges) of a graph, in such a way that knowing the symmetry group of the graph, equivalent label assignments are avoided prospectively. Recent work has been directed towards supplying the vertex-graphs needed by the generator. A program has been written to generate all graphs with t trivalent and q quadrivalent vertices, from graphs having (t + 2q) trivalent vertices. This method, however, will generate redundant vertex-graphs and some issues in isomorph elimination are considered in this presentation.

i

DD FORM 1 NOV 68 1473 (PAGE 1)

9101-807-6801

Unclassified

Security Classification

A. INTRODUCTION

The CYCLIC STRUCTURE GENERATOR forms the heart of a large family of programs that constitute an application of artificial intelligence to problems of chemical structure inference [1]. The early development of the programs utilized a systematic generator of all acyclic [topologically tree-like] chemical isomers consistent with a specified chemical composition [2 and 3]. The range of interesting chemical problems that could be solved were limited by the acyclic character of the structure generator. Recently a generator for the complete space of all cyclic and acyclic molecules has been completed [4]. The present article concerns itself with the basis set of vertex graphs[1] that the cyclic structure generator draws upon. It will be helpful however to preface this work with a brief description of the cyclic structure generator.

The problem posed to the cyclic structure generator can be described in non-chemical terms as follows: Given a sequence of numbers (A1,A2,A3...) algorithmically construct a representative set of the distinct isomorphism classes of connected, loop-free graphs having A1 vertices of valence (or degree) 1, A2 vertices of valence 2 and so on. A machine implementation of a reasonably efficient algorithm has been presented previously [4]. The algorithm has been shown to generate a complete set of graphs with anticipatory avoidance of redundancies, thereby obviating isomorphism checking.

---

(1) See section C for definition.

B. BACKGROUND ON THE CYCLIC STRUCTURE GENERATOR

One way of conceptualizing a generator is by describing it as a transformation or a mapping, T, from a basis set, B, to the generated set, G.

$$B \xrightarrow{T} G$$

Often the transformation is many-to-one from B to G, giving rise to repeated generation of several elements in G. Consequently, one faces the possible problem of removing redundancies from the generated set. Sometimes, not every element in B leads to a valid element in G. In that case one is faced with an incomplete generation of G and/or the task of processing or detecting unfruitful elements in B. Often a scheme of generation maps each element of B into a subset of G. A desirable characteristic then is that each such subset be disjoint from every other.

The cyclic generator is a composite of transformations (illustrated in Figure 1).

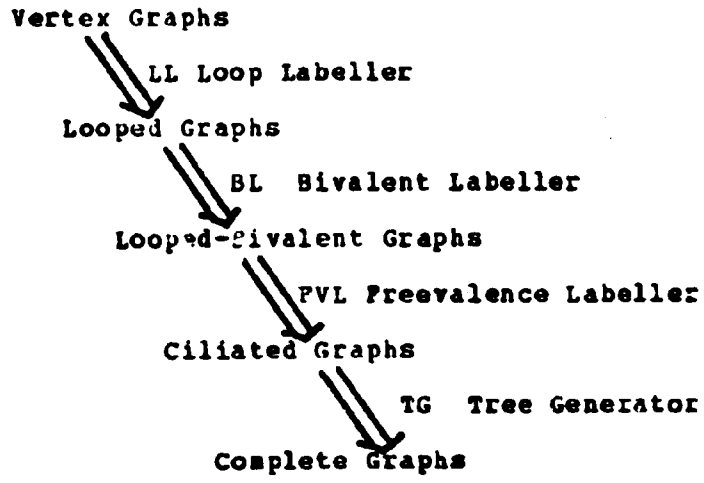


Figure 1.

The transformations begin with a class of vertex graphs which are loop-free and 2-edge-connected. (e.g., Figure 2a)

Step 1. Loop Labeller.

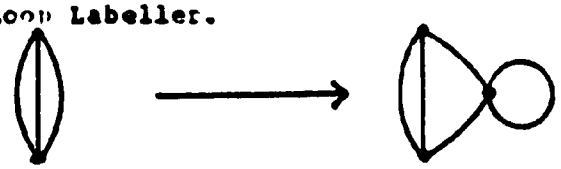


Figure 2a

This is a one-to-many transformation (possibly degenerate) designed in such a way that each vertex graph will yield a disjoint subset of looped graphs.

Step 2. Bivalent Labeller

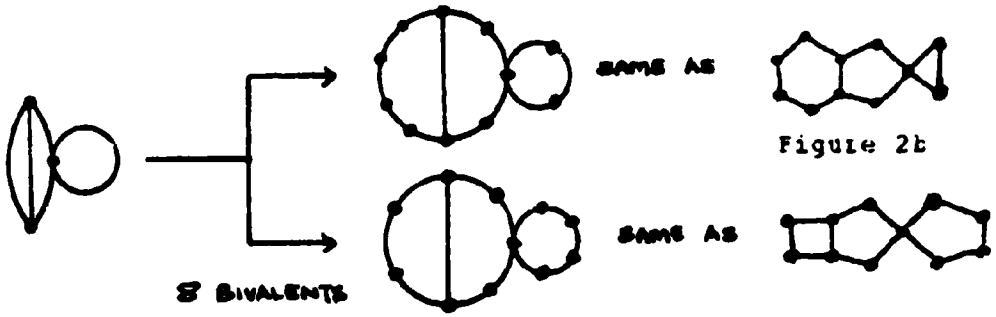


Figure 2b

This one-to-many transformation inserts vertices on edges of looped graphs in such a way that each looped graph results in a disjoint subset of looped-bivalent graphs.

Note. The first two steps may be performed repeatedly, adding loops on loops subject to certain constraints.

Step 3. The freevalence labeller designates unique ways of selecting points of attachment on the looped-bivalent graphs. These points of attachment will be used to interconnect these ciliated graphs in tree structures.

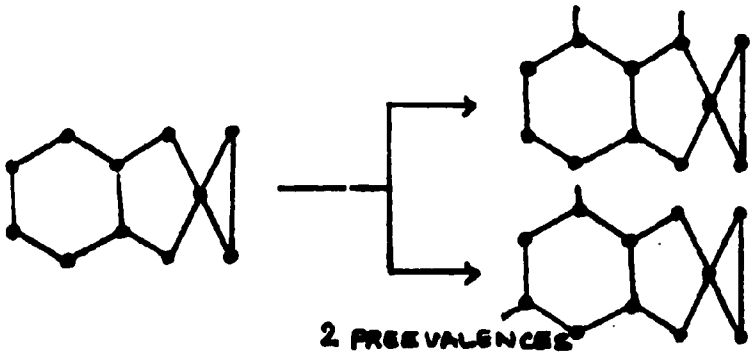


Figure 2c

Here again this one-to-many transformation produces disjoint subsets.

Step 4. The tree generator produces tree-like connected structures from a combination of ciliated graphs. Single vertices are introduced in the tree as degenerate ciliated graphs.

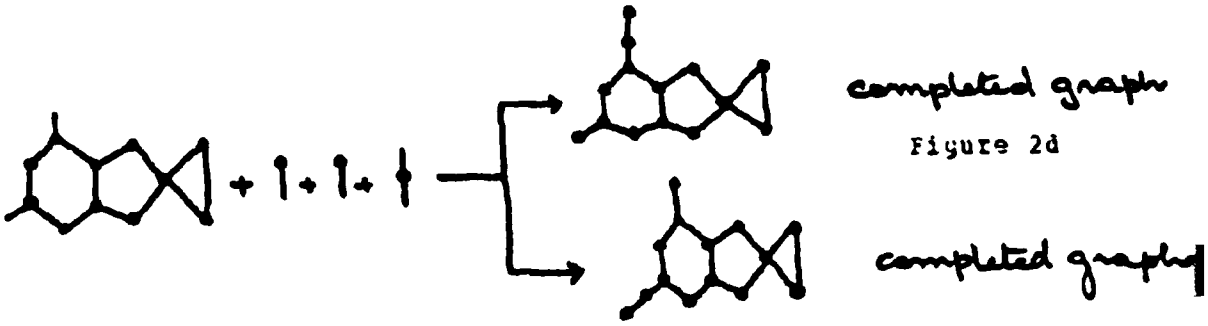


Figure 2d

The tree generator is a many-to-many transformation, where again the subsets of complete graphs produced are mutually disjoint.

Completeness & Irredundancy. In order to demonstrate the completeness and irredundency of the generated set for any scheme of generation, one needs to show that

a) If the basis set is complete and irredundant, then the transformation will yield the complete, irredundant generated set.

b) The basis set is complete and irredundant.

Frequently, the proof of (b), however trivial, is only given implicitly. In such cases, the basis set is a set whose characteristics regarding (b) are well known, e.g., the set of positive natural numbers P(N) up to N.

It has been shown [4] that (a) is true for all the distinct steps in the cyclic structure generator. Consequently, if the set of vertex graphs given to the cyclic structure generator is complete and irredundant then the algorithm will generate a complete irredundant list of isomers. In other words, the cyclic structure generator is only as complete as the set of vertex-graphs provided.

It is hoped that this brief explanation has served to place the main problem addressed in this paper in the proper perspective.

C. VERTEX-GRAPH GENERATION

The vertex-graphs serve as the basis set for the generation of chemical graphs. The cyclic structure generator builds upon each graph in the basis set by

a) adding vertex self loops where appropriate

b) inserting additional vertices of valence 2

c) constructing 1-connected structures, as appropriate, by embedding the 2-connected graphs in rooted trees.

This serves to define clearly the requirements on the basis set. Every vertex-graph should

a) have vertices of valence 3 or higher

b) be 2-connected

c) have no self-loops.

Multi-graphs are implied.

Interest in organic chemistry helps to confine attention to graphs with a maximum vertex valence of 4*.

---

* Carbon the most abundant atom in organic molecules has a valence 4. The cyclic structure generator can embed atoms of any valence in trees. That covers all cases of interest in atoms of valence higher than four. Ring atoms of valence higher than four are rare.

We shall represent graphs with t vertices of valence 3 and q vertices of valence 4 as G(t,q).

Trivalent Regular Graphs.

All graphs G(t,0) are listed by Lederberg [5], for values of t = 2,4,... up to 18. (It is easy to see that t has to be even, since the sum of vertex valences has to be even). His listings include polygonal (possessing Hamilton circuit) as well as non-polygonal graphs, and are complete. He also presents a notational system that will accommodate any polyhedron that has a Hamilton circuit, as well as unions of such polyhedra.

D. GENERATION OF GRAPHS WITH QUADIVALENT VERTICES.

The method of vertex pair promotion is presented here as a bootstrap procedure to generate G(t,q) graphs using Lederberg's listing of G(t,0) graphs.

D1. Method of Vertex Pair Promotion.

Lederberg [5] has proposed that a 4-valent vertex (a) can be treated suitably as the collapse of a pair of connected 3-valent vertices (b).

(a) (b)

Figure 3

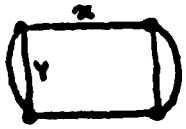
Thus the scheme involves identifying pairs of 3-valent vertices that are connected, and promoting them to 4-valent vertices by a process of collapsing the edges. It is clear that in one organization all graphs G(t,q) can be constructed in a sequence of q steps starting with graphs G(t+2q,0) and successively deriving G(t+2q-2,1), G(t+2q-4,2) ... G(t,q). The q edges that are collapsed in any G(t+2q,0) graph should be vertex-disjoint. Hence a set of q edges can be collapsed in q! sequences in the above method, yielding a maximum redundancy of q! in the list of generated graphs. The following variation attempts to avoid this redundancy, and derives benefit from knowing the symmetry group of the G(t+2q,0) graphs.

In this method, q edges are selected in unique ways from a G(t+2q,0) vertex graph, T, by a process of Edge-labelling. (For a discussion of Edge-labeling see Appendix A). The edge labelling assures us that the edges selected are unique with due consideration to the symmetry of the graph T.

Example

To generate: G(2,1)

We consider the two graphs G(4,0)



Graph 4AA

Graph 4BB

Graph 4BB has the full symmetric group on its edges. Thus, according to edge-labelling there is only one possible choice of one edge in 4bb. Upon vertex pair promotion this results in g1(2,1)

g1(2,1)

Edge labelling on 4AA yields two choices of a single edge marked x and y. Upon promoting the vertex pair x one gets the graph g1(2,1) again. The vertex pair y however gives the following graph.

g2(2,1)

The resulting graph has a self-loop. Self-loops are forbidden on vertex graphs because the cyclic structure generator synthesizes looped graphs from the vertex graphs.

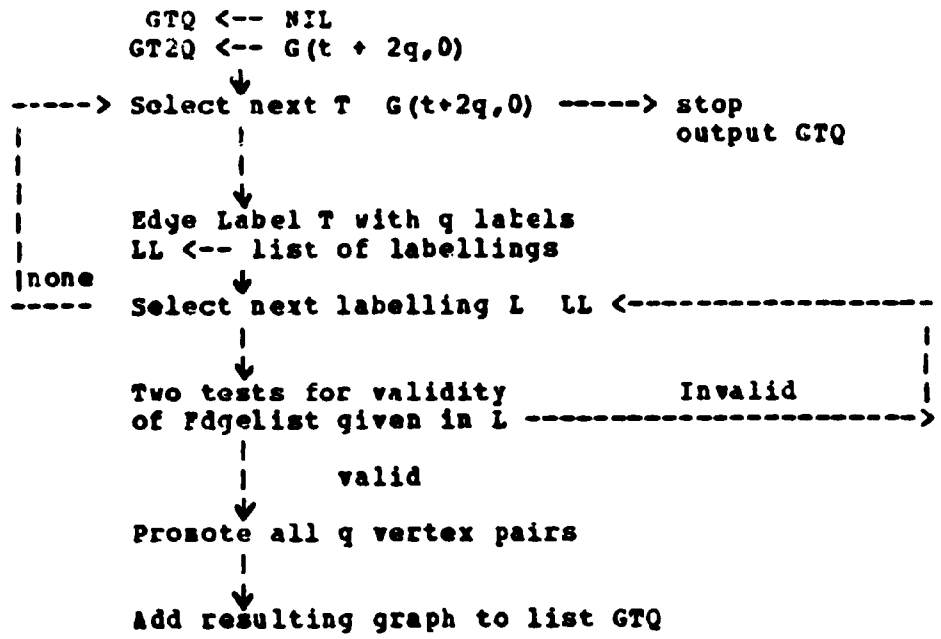
The following two validity checks are made on each edgelist produced by edge labelling

a) the edges must be vertex disjoint

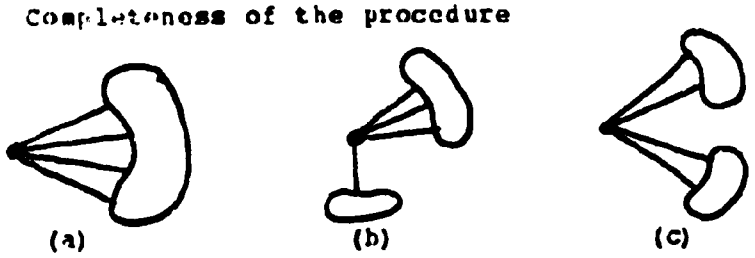
b) no vertex pair chosen can have multiple edges connecting it.

As was shown above, the resulting list of G(t,q) graphs can have isomorph redundancies that need to be eliminated. Some techniques are reviewed in Section E.

The flowchart for this method of generation is given below.



D2. Completeness of the procedure



The above configurations exhaust the possibilities for a 4-valent vertex, where the blobs represent connected parts of the graph.

Case (b) falls outside the realm of required generation, being one-connected.

Case (a) is completely covered by our generation procedure of vertex pair promotion.

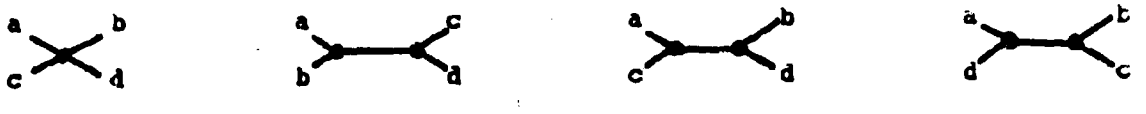
Case (c) represents a vertex which is also a cutnode and falls within the scope of required generation.

Both (e) and (f) can lead to case (c). Our method will generate case (c) from 2-connected graphs, like (f). Thus it will not be necessary to consider any 1-connected graph for the basis set. In this manner, the scheme generates the complete range of required graphs.

E. ISOMORPH ELIMINATION TECHNIQUES

E1. Origin of the Redundancy.

In the vertex pair promotion method of generation of vertex graphs, a 4-valent vertex (a) could have been generated from any one of three possible divisions of its incident edges (b,c,d).



Thus there is a maximum of 3q-fold redundancy possible for any one G(t,q) graph. However, in practice the redundancy factor is much smaller owing to the symmetry of the G(t+2q,0) graphs.

E2. Graph Isomorph Checking.

There are several excellent algorithms that are designed for testing isomorphism of a pair of graphs [16]. We have adopted a slightly modified version of an algorithm designed by Buchanan [unpublished]. There is no proven assertion about the efficiency of this algorithm but it has heuristic merit. The algorithm is oriented toward detecting non-isomorphism quickly.

A general comment on methods of isomorphism checking is in order. Most methods, to our knowledge, make no utilization of the symmetry of the graphs under comparison. Consider, for example, testing a pair of graphs, A against B. Let B have a high order of symmetry. The algorithm repeatedly may initiate a match for vertex a of A with vertex b of B. When further excursions into matching reveal that a cannot be matched with b, the algorithm may try another vertex of B, say b1 which can be in the orbit of symmetry of vertex b. We know from the automorphisms of graph B that if b is invalid as match for a, then so is any other vertex in the orbit of b. Thus the symmetry group can be employed fruitfully during isomorph checking. It is worthwhile to compare some characteristics of the groups of the two

There are no labels associated with the vertices. Hence, the features (i) through (iv) are ineffective for grouping. The following features are used:

a) a sequence representing the number of multiple edges

b) the number of 3-cycles; generalizable to a list of up to k-cycles

c) the eigenvalues (*) of the adjacency matrix of the graph,

   if (a) and (b) leave large sets of graphs unresolved.

(*) see Appendix B.

---

graphs before embarking on the search for the isomorphism mapping.

E1. Grouping the list of graphs.

When a list of graphs, L, is to be processed for isomorph elimination, the worst case (each graph is unique) will engender (n*n-n)/2 tests of isomorphism. When isomorphs are present in L, this number is reduced considerably when care is taken to test each new candidate with a list of unique graphs alone.

A technique used to expedite detection of non-isomorphism of a pair of graphs involves comparing certain easily computed topologically invariant features of the graphs. If the pair of graphs do not correspond in all their features, non-isomorphism can be pronounced readily . Extending this to the isomorph elimination from a list of graphs, L, leads to the grouping technique. Grouping involves a pass through L and associating with each graph its computed features. This list L then can be organized into sublists such that each pair of graphs in a sublist has identical features. Though a simple technique, grouping leads to considerable reduction in the number of isomorphism tests to be made.

Choice of features to use in grouping:

Usual features that are checked before isomorph testing include the

i) number of vertices,

ii) number of edges,

iii) the vertex-valence list,

iv) a sequence listing the labels associated with vertices, if any, and

v) a sequence listing the labels on edges if any.

In our application it is guaranteed that all graphs in a list L to be processed for redundancy have

a) the same number of vertices

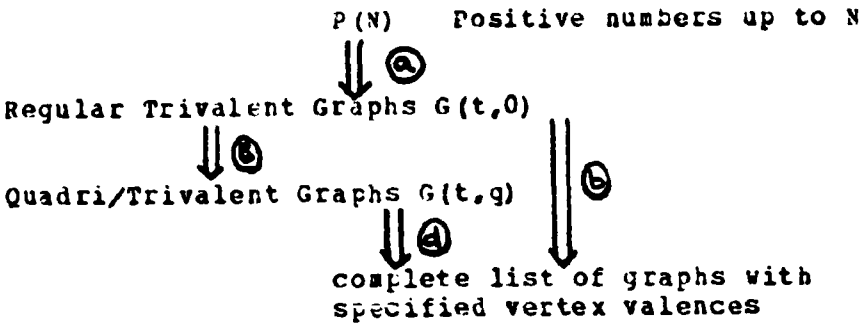
b) the same number of edges

c) the same vertex-valence list.

E4. Use of Canonical Forms to Avoid Graph Matching.

Morgan (*) and Heap (*) give procedures to represent graphs canonically. These procedures can lead to sizeable expenditure of computing effort, especially in the presence of symmetry in the graphs. Lederberg (*) also has devised a canonical notation useful in describing the graphs we generate. However, these methods are not used in the present scheme.

F. Conclusion



The generation (b) of complete lists of chemical graphs compatible with specified atomic compositions (valence lists) was hindered originally by the incompleteness of the available list of Quadri/trivalent graphs. The regular trivalent graphs G(t,0) to be included in the basis set for generation (b) were generated themselves in an exhaustive way (a). The present paper has described an implementation of a suggestion by Lederberg for bootstrapped generation (c) of Quadri/trivalent graphs from regular trivalent graphs. A separate report lists G(t,q) graphs for several values of t and q. [7]

Appendix A. Edge-Labelling.

The general "labelling" problem has been given a group theoretic definition, analysis and solution by Brown, Masinter and Hjelmeland [9]. In the present context of selecting unique sets of k edges from a graph G, a reduced formulation of the labelling problem will suffice. Let G be a graph with an arbitrary definition of indexing its edges from 1 through m. Let L be the label set, with k labels of one type and (m-k) of another type. We shall denote the automorphism group of G represented as permutations on the indices of its edges, by $\mathcal{G}$. The label set admits a group $\mathcal{L}$ Since each label is indistinguishable from other labels of the same type, $\mathcal{L}$ can be written as the direct sum $S(k) + S(m-k)$. The indexed labellings of the edges of G by L can be identified with S(m) the full permutation group. Two labellings $\pi_1, \pi_2 \in S(m)$ are equivalent when $\pi_2 = g\pi_1 l$ for any $g \in \mathcal{G}$ and $l \in \mathcal{L}$. The equivalence class determined by

$$\{g\pi l \mid g \in \mathcal{G}, l \in \mathcal{L}\} \triangleq \mathcal{G}\pi\mathcal{L}$$

is called a double coset*

---

* analogous to $\{\alpha\pi \mid \alpha \in A\} \triangleq A\pi$
or $\{\pi\alpha \mid \alpha \in A\} \triangleq \pi A$
being called single cosets. See [8].

The groups $\mathcal{G}$ and $\mathcal{L}$ induce a partitioning of S(m), by means of their double cosets. A set of double coset representatives constitutes a complete set of unique labellings of G with labels L.

Brown, Masinter and Hjelmeland [9] have designed and implemented two algorithms for generating double coset representatives for the labelling problem. The algorithm used in the present work was an implementation given by L. Masinter.

Appendix 3. Eigenvalues as Invariant Feature
(by Dr. Raymond Carhart)

A graph can be represented as adjacency matrix A of order nxn, where n is the number of vertices in the graph. Each Aij is an integer representing the number of edges linking vertices i and j. (Aij is zero when vertices i and j are not connected by an edge). For an undirected graph the matrix A is symmetric. When there are no self-loops on vertices the diagonal elements Aii are zero. Since simultaneous row and column permutations on A leave the topology of the graph unaffected, two graphs (matrices) A and B are topologically equivalent if they are related by a permutation matrix P such that

$$A = P^T B P \quad \text{or} \quad A \overset{P}{\longleftrightarrow} B \qquad \text{----------(C1)}$$

The eigenvalues of a symmetric matrix A can be defined as the diagonal elements of a diagonal matrix Λ, such that

$$\Lambda = U^T A U \qquad \text{.....(c2)}$$

where U is any orthogonal matrix.

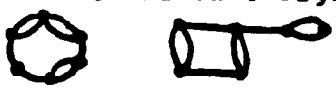
Substituting (c1) into (c2) we find

$$= (PU)^T B (PU) \qquad \text{------ (c3)}$$

When U is orthogonal, (PU) is orthogonal as well. Thus the eigenvalue sets of the adjacency matrices of topologically equivalent graphs are the same.

$$A \overset{P}{\longleftrightarrow} B \Longrightarrow \Lambda(A) = \Lambda(B).$$

In general it is not true that when the eigenvalues of two adjacency matrices are the same, the two corresponding graphs are topologically equivalent. For example, the two graphs shown below are topologically different, but have the same eigenvalue sets.



The eigenvalue computation can be used as an invariant feature or topologically equivalent graphs to aid in the grouping process

preceding the isomorph elimination process.

Table I. SUMMARY OF RESULTS

Vertex Graphs with Trivalents	Vertex Graphs with Quadrivalents	Number of Graphs Generated	Generated from Trivalent Regular Graphs of
2	1	1	4 vertices
4	1	5	6 vertices
6	1	24	8 vertices
0	2	1	4 vertices
2	2	4	6 vertices
4	2	34	8 vertices
0	3	1	6 vertices
2	3	12	8 vertices
0	4	3	8 vertices

See Reference [7] for complete lists of graphs.

References

[1] B.G. Buchanan and J. Lederberg, "The Heuristic DENDRAL Program for Explaining Empirical Data". Proceedings of the IFIP Congress 71, Ljubljana, Yugoslavia (1971).

[2] J. Lederberg, "DENDRAL-64" Part I. Notational Algorithm for Tree Structures. NASA Technical Report CR 57029, 1964.

[3] J. Lederberg, G.L. Sutherland, B.G. Buchanan, E.A. Feigenbaum, A.V. Robertson, A.M. Duffield and C. Djerassi, "Applications of Artificial Intelligence for Chemical Inference I. The Number of Possible Organic Compounds: Acyclic Structures containing C, H, O and N". Journal of the American Chemical Society, 91:11 (May 21, 1969).

[4] H. Brown, L. Masinter, "An Algorithm for the Construction of the Graphs of Organic Molecules", submitted to Discrete Mathematics, 1973.

[5] J. Lederberg, "DENDRAL-64" Part II. Topology of Cyclic Graphs. NASA Technical Report CR 68898, 1965.

[6] J. Hopcroft and R. Tarjan, "A V^2 Algorithm for Determining Isomorphism of Planar Graphs". Information Processing Letters, Vol. 1, No. 1, 1971.

[7] N. S. Sridharan, "A Catalog of Quadri/Trivalent Graphs", in preparation, 1973.

[8] E. Beckenbach, Applied Combinatorial Mathematics, Wiley, New York, 1964.

[9] H. Brown, L. Masinter and L. Hjelmeland, "Constructive Graph Labelling Using Double Cosets", to appear in Discrete Mathematics, 1973.