(8) Alcantara, R. B.; Westerberg, A. W.; Rychener, M. D. *Comput. Chem. Eng.* **1985**, *9*, 127-142.
(9) Kelly, E. B.; Holste, J. C.; Hall, K. R. AIChE 1987 Annual Meeting, Session 138.
(10) Gani, R.; O'Connell, J. P. *Comput. Chem. Eng.* **1989**, *13*, 397-404.
(11) Dong, Q.; Xu, Z.; Wang, P. *Anal. Chim. Acta* **1988**, *210*, 181-187.
(12) Huang, G. Doctoral Thesis, the Institute of Chemical Metallurgy, Academia Sinica, Beijing 100080, China, 1989.
(13) Xu, Z.; Sun, R.; Dong, Q.; Yan, X. *Comput. Appl. Chem. (China)* **1989**, *6* (2), 1-5.
(14) *VAX/VMS Run Time Library Reference*; Documentation Publishing Center of 2000 Series: Beijing, 1987.
(15) *VAX Rdb/VMS database design and definition*; Documentation Publishing Center of 2000 Series: Beijing, 1987.
(16) Guo, L., et al. A database subsystem of KB-OPDS. In *KB-OPDS*, Research Report to the Chinese Academy of Sciences and the Ministry of Chemical Industry, China (Aug 1989) pp 21-40. Available from the Institute of Chemical Metallurgy, Academia Sinica, P.O. Box 353, Beijing, China.
(17) Reid, R. C., et al. *The Properties of Gases and Liquids*, 4th ed.; McGraw-Hill: New York, 1987.
(18) Stephenson, R. M.; Malanowski, S. *Handbook of the Thermodynamics of Organic Compounds*; Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1987.
(19) Simmrock, K. H.; Janowsky, R.; Ohnsorge, A. *Critical Data of Pure Substances (Ag - C7)*; Chemistry Data Series; DECHEMA: Frankfurt, 1986; Vol. II, Part 1.
(20) Smith, B. D.; Srivastava, R. *Thermodynamic Data for Pure Compounds*; Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1986, Parts A and B.
(21) Li, X., et al. A Prediction Subsystem of KB-OPDS. In KB-OPDS, Research Report to the Chinese Academy of Sciences and the Ministry of Chemical Industry, China (Aug 1989) pp 41-68. Available from the Institute of Chemical Metallurgy, Academia Sinica, P.O. Box 353, Beijing, China.
(22) Dong, Q., et al. A Generalized Subsystem of Automation of Group Additive Methods. In *KB-OPDS*, Research Report to the Chinese Academy of Sciences and the Ministry of Chemical Industry, China (Aug 1989) pp 70-86. Available from the Institute of Chemical Metallurgy, Academia Sinica, P.O. Box 353, Beijing, China.
(23) Yan, X., et al. A Subsystem for Organic Structure Information Processing. In *KB-OPDS*, Research Report to the Chinese Academy of Sciences and the Ministry of Chemical Industry, China (Aug 1989) pp 88-125. Available from the Institute of Chemical Metallurgy, Academia Sinica, P.O. Box 353, Beijing, China.
(24) King, R. B.; Rouvray, D. H. *Graph Theory and Topology in Chemistry*; Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1987.
(25) Li, X., et al. Systematic Qualification of data and Models. In *KB-O-PDS* Research Report to the Chinese Academy of Sciences and the Ministry of Chemical Industry, China (Aug 1989) pp 126-147. Available from the Institute of Chemical Metallurgy, Academia Sinica, P.O. Box 353, Beijing, China.
(26) *VAX Digital Command Language Reference*; Documentation Publishing Center of 2000 Series: Beijing, China, 1987.

Computational Techniques for Vertex Partitioning of Graphs

XIAOYU LIU, K. BALASUBRAMANIAN,*,[‡] and M. E. MUNK*

Department of Chemistry, Arizona State University, Tempe, Arizona 85287-1604

Received March 5, 1990

A powerful vertex-partitioning algorithm is developed and applied for vertex partitioning of graphs of chemical and spectroscopic interest. The codes developed on the basis of these algorithms are tested and compared for performance with other methods based on the Morgan algorithm and the principal eigenvector algorithm based on the Givens-Householder method. The newly developed algorithm and codes appear to be more powerful than the Morgan and the principal eigenvector algorithms for vertex partitioning of graphs.

INTRODUCTION

The graph-vertex-automorphism partitioning problem has received considerable attention in recent years.[1-12] The vertex partitioning is of fundamental interest since it has many practical applications. First and foremost of all, it provides a solution for computer perception of the hidden topological symmetry of a molecule. In our group we have been interested in building a comprehensive computer-assisted structure-elucidation system.[11,12] A critical problem encountered in this work is that given the neighborhood table (equivalently the adjacency matrix) of a molecule, can an automated algorithm and code be written to yield its topological symmetry.

The graph-vertex-partitioning problem also finds important application in the computer generation of ^{13}C and proton NMR signals of molecules and their intensity patterns. We have described in an earlier manuscript the application of the vertex partitioning to generate ^{13}C NMR spectra.[24]

Some early techniques for generating vertex partitioning of graphs have been based on the Morgan algorithm.[1-5] and the principal eigenvector algorithm. Randić and co-workers[10] have formulated the canonical vertex-labeling method to generate the vertex-automorphism partitionings of graphs. Although many of these algorithms and the codes based on these algorithms are simple to use, they often lead to convergence problems and oscillatory behaviors. Herndon and co-workers[6-8,25] have discussed these aspects in considerable depth. These algorithms, including the canonical labeling technique, often do not provide satisfactory solutions for highly transitive graphs, viz., those graphs which contain many vertices of the same degree. For graphs which contain several isospectral points (two nonequivalent vertices in the graph which yield isospectral graphs if identical fragments are attached to these vertices), direct application of both the Morgan algorithm and the principal eigenvector schemes do not provide correct solutions.

Shelley and Munk[11] have developed an extended Morgan algorithm approach for vertex partitioning of graphs. As discussed in the literature (see ref 25), this algorithm cannot discriminate between isospectral points. However, a revised procedure proposed by Shelley and Munk[12] subsequently circumvents this problem. In their second paper, Shelley and Munk have used procedures to construct the sequence number permutations in the early stage of their algorithm. A combination of this procedure and the earlier methods was shown to construct the automorphism partitioning of many graphs, including graphs which contain isospectral points. This second paper of Shelley and Munk[12] has apparently been overlooked in the literature.

In this investigation we develop and apply algorithms and codes for vertex partitioning of graphs. We also critically compare the performance of newly developed codes with the codes based on Morgan, principal eigenvector, and other algorithms. It is demonstrated that the codes developed here

[‡]Camille and Henry Dreyfus Teacher-scholar.

provide satisfactory solutions to graphs containing isospectral points and transitive graphs.

MATHEMATICAL PRELIMINARIES AND ALGORITHMS

Given a chemical graph $G(V,E)$, where V denotes the vertex set and E the edge set, its adjacency matrix is defined as

$$\mathbf{A} = (a_{i,j}) = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases} \qquad (1)$$
$$i,j = 1, 2, ..., |V| = n$$

An automorphism of a graph G is an isomorphism of G with itself. Thus each automorphism of G is a permutation of the vertex set V, which preserves the adjacency. Mathematically, if $\mathbf{P}$ is an $n \times n$ permutation matrix which corresponds to a permutation of the vertices of the graph, then $\mathbf{P}$ belongs to the automorphism group if and only if

$$\mathbf{P}^{-1}\mathbf{A}\mathbf{P} = \mathbf{A} \qquad (2)$$

The set of all such automorphisms of a graph G under the binary operation of composition is always a group, called the automorphism group, and is denoted by Γ. A partitioning of the vertex set V of the graph G is called the automorphism partitioning, denoted by $\mathbf{P}_\Gamma$, when the following holds: vertices i and j belong to the same class (or cell) if and only if there exists at least one automorphism of G mapping i onto j, that is

$$\pi(i) = j, \qquad \pi \in \Gamma \qquad (3)$$

The number of classes (or subsets) under the automorphism partitioning is denoted by n_Γ.

We associate with the adjacency matrix A of graph G, a $n \times 1$ column vector $\mathbf{X}$, defined as

$$\mathbf{X} = (x_1, x_2, ..., x_n)^T \qquad (4)$$

such that there is a one to one correspondence between each vertex $i \in V$ and the component x_i of the vector $\mathbf{X}$. A partitioning of the vertex set V is then called the vector partitioning (or vertex partitioning), denoted by $\mathbf{P}_V$, when the following holds: vertices i and j belong to the same class if and only if $x_i = x_j$. Let n_V be the number of different x_i's of the vector $\mathbf{X}$. The vector partitioning defined above then decomposes the set V into n_V disjoint subsets, namely

$$\mathbf{V} = \mathbf{V}_1 \cup \mathbf{V}_2 \cup ... \cup \mathbf{V}_{n_V} = \bigcup_{i=1}^{n_V} \mathbf{V}_i$$
$$\mathbf{V}_i \cap \mathbf{V}_j = \phi \text{ for } i \neq j \qquad (5)$$

Let I be the index set that collects the vertices which have been partitioned. The vector-partitioning technique can be cast into a pseudocode module shown in Table I. Clearly, generation of the vector $\mathbf{X}$ is essential to the vector-partitioning process. Some of these are well-known results which are reviewed for the sake of making our discussions self-explanatory. We first discuss below some well-known vector-generating algorithms. The following lay the foundations for the Morgan and the principal eigenvector algorithms, respectively.

Theorem 1. In the kth matrical product, $\mathbf{A}^k$, of the adjacency matrix, defined inductively as follows:

$$\mathbf{A}^k = \mathbf{A}^{k-1}\cdot\mathbf{A} \qquad (\mathbf{A}^0 = I) \qquad (6)$$

a_{ij}^k is the number of walks from i to j, containing k edges; the sum, denoted by c_i, of these elements a_{ij}^k over all values of j (i.e., the sum of the ith row) gives the number of all possible walks of the length k from vertex i to all other vertices in a (connected) graph.[13]

Table I. Vector-Partition Algorithm

```
    Module Vect_Partitioning
    Get vector $\mathbf{X} = (x_1, x_2, ..., x_n)^T$
    $I = \phi$; $k = 0$ !Set I contains the checked vertices
    For $i = 1, n$, do
      If ($i \in I$) Go to 20
      $k = k + 1$
      $V_k = V_k \cup \{\lambda\}$
      For $j = i + 1, n$, do
        If ($j \in I$) Go to 10
        If ($x_i = x_j$) then
          $V_k = K_k \cup \{\theta\}$
          $I = I \cup \{j\}$
        End if
10    End do
      $I = I \cup \{i\}$
20  End do
    $n_V = k$
    Output subsets $V_i$, $i = 1, 2, ..., n_V$
    End module
```

Theorem 2. If λ_1 is the principal eigenvalue (the largest eigenvalue in magnitude) of the adjacency matrix $\mathbf{A}$, then the column vector $\mathbf{U}_k$, generated by the iterative procedure

$$\mathbf{U}_k = \mathbf{A}\cdot\mathbf{U}_{k-1} \text{ [with } \mathbf{U}_0 = (1, 1, ..., 1)^T]$$
$$\mathbf{U}_k = \mathbf{U}_k/\max_i(u_i)_k \qquad (7)$$

is known to converge upon the principal eigenvector $\mathbf{U}$ of λ_1 as $k = \infty$.[14,15]

While working on a machine description of chemical structures, Morgan[1] first introduced c_i's defined above as "extended connectivity values". By using the extended connectivity value, a piece of adjacency information, Morgan described what is now known as the Morgan algorithm: assign to each vertex an initial "connectivity value" equal to the degree of the vertex. The number (n_V) of different "connectivity values" which had been assigned is then calculated. The next step assigns a new extended connectivity value for each vertex by summing the previous extended connectivity values of adjacent vertices. The number (n_V') of different "connectivity values" is then calculated. If $n_V' > n_V$, then set n_V is equal to set n_V', and the summation process is repeated. If, however, after $k + 1$ iterations, $n_V' \leq n_V$, the process is terminated, and the last set of values is assigned to the vertices and used to induce a partitioning among the vertices.

Razinger[5] later showed theoretically that Morgan's extended connectivity value for the vertex i in the kth iteration is simply c_i, the number of all possible walks of length k from the vertex i to all other vertices in a given graph, which, as we know, is well defined by Theorem 1. Thus, if the convergence criterion is set up as

$$\text{if } n_V' \leq n_V \text{ then stop} \qquad (8)$$

Based on Theorem 1 above, Morgan's vector-generating (i.e., extended connectivity values) algorithm can be formalized as a pseudocode module shown in Table II. The vector $\mathbf{C} = (c_1, c_2, ..., c_n)^T$ is then used to induce a partitioning among the vertices.

The well-known Theorem 2 offers a simple iterative procedure to calculate the principal eigenvector. Using the convergence criterion (eq 8), where n_V is the number of different components of the vector $\mathbf{U}_k$, another vector-generating algorithm can then be devised from Theorem 2, given in Table III. The $\mathbf{U}_k$ vector generated by using the algorithm in Table III can be used to induce a partitioning among vertices. It should be noted that it is not necessary for the convergence criterion (eq 8) to guarantee that the vector $\mathbf{U}_k$ has converged upon the principal eigenvector $\mathbf{U}$ due to the fact that the condition $\mathbf{U}_k = \mathbf{U}$ is only realized for $k = \infty$.

Next we describe a more time-consuming and complicated but somewhat more deterministic numerical method to gen-

Table II. Vector-Generator I Algorithm

```
    Module Vect_Generator_I
    Get $n$, $\mathbf{A} = (a_{ij})$, $i, j = 1, ..., n$ !A the adjacency matrix
    $n_V = 0$; $k = 1$
111 Compute $\mathbf{A}^k = \mathbf{A}^{k-1}\mathbf{A}$
    Construct $\mathbf{C} = (c_1, c_2, ..., c_n)^T$ such that
        $c_i = \sum_{j=1}^{n} a_{ij}^k$
        $i = 1, 2, ..., n$
    Compute $n_V'$
    If ($n_V' > n_V$) then
        $n_V = n_V'$
        $k = k + 1$
        Go to 111
    End if
    If ($n_V' \leq n_V$) then
        Construct $\mathbf{C} = (c_1, c_2, ..., c_n)^T$ such that
            $c_i = \sum_{j=1}^{n} a_{ij}^{k-1}$
            $i = 1, 2, ..., n$
    End if
    Output $n_V$, $\mathbf{C} = (c_1, c_2, ..., c_n)^T$
    End module
```

Table III. Vector-Generator II Algorithm

```
    Module Vect_Generator_II
    Get $n$, $\mathbf{A} = (a_{ij})$, $i, j = 1, ..., n$ !A the adjacency matrix
    $n_V = 0$; $k = 1$
    Let $\mathbf{U}_0 = (1, 1, ..., 1)^T$
111 Compute $\mathbf{U}_k = \mathbf{A}\,\mathbf{U}_{k-1}$
    Compute $n_V'$
    If ($n_V' > n_V$) then
        $n_V = n_V'$
        $k = k + 1$
        Go to 111
    End if
    Output $n_V$, $\mathbf{U}_k = (u_{k1}, u_{k2}, ..., u_{kn})^T$
    End module
```

erate the principal eigenvector **U**. We employ the Givens–Householder method[16-20] to calculate the spectrum of a given graph. The principal eigenvalue is then simply defined as

$$\lambda = \max_i\{|\lambda_i|\} \tag{9}$$

The corresponding principal eigenvector **U** is then evaluated by solving the equation

$$\mathbf{A}\cdot\mathbf{U} = \lambda\cdot\mathbf{U} \tag{10}$$

in which the adjacency matrix **A** has been brought into a tridiagonal form by the linear orthogonality transformation proposed by Givens and Householder. Table IV shows the third vector-generating algorithm based on the Givens–Householder method. The principal eigenvector **U** generated by the algorithm in Table IV can be used to induce a partitioning among vertices. The reader is referred to refs 16–20 for a detailed description of the Givens–Householder method.

All three algorithms presented above (Tables II–IV) generate the vectors

$$\mathbf{C} = (c_1, c_2, ..., c_n) \tag{11}$$

$$\mathbf{U}_k = (u_{k_1}, u_{k_2}, ..., u_{k_n}) \tag{12}$$

and

$$\mathbf{U} = (u_1, u_2, ..., u_n) \tag{13}$$

Each of these vectors can be called by the vector-partitioning module in Table I to induce the corresponding vector partitioning among vertices.

The final algorithm we discuss is different from the approaches given above in a sense that instead of associating each vertex i with a component x_i of a vector **X** we now assign the

Table IV. Vector-Generator III Algorithm

```
Module Vector_Generator_III
Get $n$, $\mathbf{A} = (a_{ij})$, $i, j = 1, ..., n$ !A the adjacency matrix
Tridiagonalize $\mathbf{A}$
Compute spectrum $\{\lambda_i\}$, $i = 1, 2, ..., n$
$\lambda = \max_i\{|\lambda_i|\}$
Solve $\mathbf{A}\cdot\mathbf{U} = \lambda\cdot\mathbf{U}$
Output $\mathbf{U} = (u_1, u_2, ..., u_n)^T$
End module
```

index i to a row vector $\mathbf{L}_i$ (or simply called a list) of certain length and is stated as follows: Assume that we have an ordered collection of subsets of V, denoted by $V_1, V_2, ..., V_K$, depending on some graph invariant or invariants (such as the degree partitioning). To each vertex i assign a list $\mathbf{L}_i = (a_1, a_2, ..., a_k)$ where a_j $(1 \leq j \leq k)$ is the number of vertices in the subset V_j adjacent to i, or more precisely a_j equals the exact number of vertices $k \in V_j$ such that $(i,k) \in E$. Using these lists, each V_j may now be partitioned into subsets, each subset consisting of all vertices with a given list. In this way we may obtain a refinement to the original collection of subsets. No refinement will be obtained if all vertices in each subset have identical lists. If a refinement has been obtained, then the method is reapplied until there is no further refinement. Corneil and Gotlieb[21] have pointed out that if the given partitioning is invariant under automorphism, the partitioning resulting from this algorithm is also invariant under that automorphism. Table V shows the iterative vertex classification algorithm in a pseudocode form. From the pseudocode in Table V, it can be seen that the two iterative steps, i.e., step 2 and step 3, are repeated until no subset is refined, that is, until the condition $t = t'$ is satisfied, and the vertex partitioning of the vertex set V of a given graph G with respect to the degree partitioning has been obtained. Under the degree partitioning two vertices belong to the same subset if and only if their degrees are the same. Therefore if G is regular then the degree partitioning consists of the vertex set V alone. Hence the two-step iteration in the algorithm in Table V does not refine the partitioning. The vectors **C**, $\mathbf{U}_k$, and **U** generated by the modules in Tables II, III, and IV, respectively, fail to induce a partitioning among vertices of a regular graph due to the fact that they are either unit vectors or can be normalized to unit vectors when regular graphs are dealt with.

Highly transitive graphs and regular graphs pose special problems. A fully transitive graph is a graph which contains a single automorphism partitioning set. For these graphs, first we used a ring-partitioning technique. Wipke and Dyott[26] have described a ring-partitioning algorithm. A combination of this with the vertex-partitioning algorithm developed here provides satisfactory solutions to vertex-transitive graphs. To each vertex of such graphs a ring list $R_i = (d_3, d_4, ..., d_m)$, where d_i is the number of element rings of size i in which the vertex occurs, is assigned. Note that m is the largest ring size present in the graph. This information is used to classify the vertices of the graph under investigation into subsets based on their ring lists. That is, it is stipulated that two vertices belong to the same subset if and only if their ring lists are identical. Subsequently, the vertex-partitioning algorithm is invoked for further refinements of vertex subsets.

In the ensuing section, we demonstrate the use of these algorithms with various graphs of chemical interests. The vertices of all the graphs considered here are randomly labeled since the equivalence classes of vertices are invariant to labeling.

RESULTS AND DISCUSSION

For many molecular graphs the vectors generated by the algorithms in Tables II and III are in general very discrimi-

Table V. Vertex-Classification Algorithm

```
      Module name: Vertex_Classification_Algorithm
      Get $n$, and adjacency matrix $\mathbf{A} = (a_{ij})$, $i, j = 1, ..., n$
      Set $V_i = \phi$, $1 \leq i \leq n$ !$V_i$'s are subsets under partition
    Step 1 Vertex partitioning
      $d_{\max} = 0$ !$d_{\max}$ is the largest vertex degree
      For $i = 1$ to $n$ do
        $d = 0$
        For $j = 1$ to $n$ do
          if $a_{ij} \neq 0$ then $d = d + 1$
        End do
        $V_d = V_d \cup \{i\}$
        If $(d > d_{\max})$ $d_{\max} = d$
      End do
      $t' = d_{\max}$
    Step 2 List generating
      $\mathbf{L}_i(j) = 0$, $1 \leq i \leq n$, $1 \leq j \leq t'$ !$\mathbf{L}_i$ is vertex $i$'s list
      For $i = 1$ to $n$ do
        For all $k \in V$, if $(k,i) \in E$ then
          For $j = 1$ to $t'$ do
            If $k \in V_j$ then $\mathbf{L}_i(j) = \mathbf{L}_i(j) + 1$
          End do
        End if
      End do
    Step 3 Subset refining
      $t = 0$
      For $i = 1$ to $t'$ do
        If $V_i = \phi$ then go to 50
        If $|V_i| = 1$ then
          $t = t + 1$; $E_t = \phi$; $E_t = E_t \cup V_i$;
          Reset $V_i = \phi$ !$E_t$'s are new subsets of $V$
        End if
501     Get $p \in V_i$; $t = t + 1$; $E_t = \phi$; $E_t = E_t \cup \{p\}$; $V_i = V_i - \{p\}$
502     Get $q \in V_i$
        If $q \neq 0$ then
          If $\mathbf{L}_p = \mathbf{L}_q$ then
            $E_t = E_t \cup \{q\}$; $V_i = V_i - \{q\}$; Go to 502
          End if
        End if
        If $V_i \neq 0$ then go to 501
50    End do
      If $t \neq t'$ then
        $t' = t$
        For $i = 1$ to $t'$ do
          $V_i = E_i$
        End do
        Go to Step 2
      End if
      End module
```

nating,[1,6,9,10] the resulting vertex partitioning under vector $\mathbf{C}$ or $\mathbf{U}_k$ is often identical with the automorphism partitioning, or at least the vertex partitioning induced by the vectors is invariant under automorphism. This case is demonstrated with graph 1 in Figure 1. From the given adjacency matrix, the vectors $\mathbf{C}$ and $\mathbf{U}_k$ are generated as

$$\mathbf{C} = (4,5,7,5,4,4,5,7,5,4)$$

$$\mathbf{U}_k = (0.5833,0.6875,1.0000,0.6875,0.5833,0.5833,0.6875,1.0000,0.6875,0.5833)$$

The induced vertex partitioning as well as the automorphism partitioning are listed as follows:

$\mathbf{P}_\Gamma$	$\mathbf{P}_V$ under $\mathbf{C}$	$\mathbf{P}_V$ under $\mathbf{U}_k$
$n_\Gamma = 3$	$n_V = 3$	$n_V = 3$
{3,8}	{3,8}	{3,8}
{1,5,6,10}	{1,5,6,10}	{1,5,6,10}
{2,4,7,9}	{2,4,7,9}	{2,4,7,9}

A difficulty that remains in both algorithms, however, is that isospectral points[6,9] are not distinguished by direct application of either procedure. Herndon has pointed out that there are numerous chemical graphs with isospectral points[6] and that

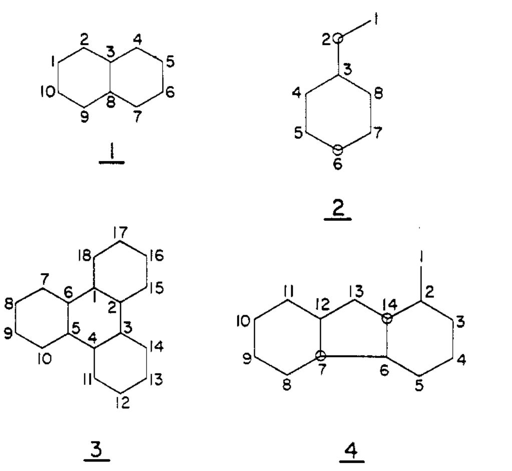

Figure 1. Four graphs used to illustrate graph-vertex-automorphism partitioning problems.

the failure of the first algorithm proposed by Shelley and Munk to generate the canonical labelings for organic compounds[11] is attributed to the presence of isospectral points. However, in their second paper[12] Shelley and Munk use a breadth-first or depth-first approach in combination with their previous algorithm[11] to obtain satisfactory solutions to many graphs, including graphs with isospectral points. For two isospectral points in pair, attachment of any moiety M to the points in turn gives two nonisomorphic graphs, which are often called isospectral graphs. Isospectral graphs are nonisomorphic graphs whose characteristic polynomials are the same and thus have the same eigenvalues. They have been the topic of many publications.[6-9,23] This is illustrated with graph 2 in Figure 1. Note that the vertices 2 and 6 are isospectral points. The vectors generated by algorithms in Tables II and III are shown as:

$$\mathbf{C} = (4,8,14,10,9,8,9,10)$$

$$\mathbf{U}_k = (0.3083,0.6241,1.0000,0.7669,0.6541,0.6241,0.6541,0.7669)$$

The vertex partitioning obtained by using the above algorithms are shown as:

$\mathbf{P}_\Gamma$	$\mathbf{P}_V$ under $\mathbf{C}$	$\mathbf{P}_V$ under $\mathbf{U}_k$
$n_\Gamma = 6$	$n_V = 5$	$n_V = 5$
{1},{2},	{1},{3},	{1},{3},
{3},{6},	{2,6},	{2,6},
{4,8},	{4,8},	{4,8},
{5,7}	{5,7}	{5,7}

Note that although the vertices 2 and 6 are not equivalent, they are classified into the same subset under the vector partitioning induced by $\mathbf{C}$ or $\mathbf{U}_k$. This indicates that for graphs with isospectral points the vertex partitioning induced by either the vector $\mathbf{C}$ or the vector $\mathbf{U}_k$ may not be invariant under automorphism. Also, the Morgan vertex-partitioning scheme exhibits nonuniform convergence and oscillatory behavior, and nonautomorphic vertices can possess the same extended connectivity values in every step.[6,10-11]

The vertex-partitioning scheme under vector $\mathbf{U}_k$ is infested with such problems as oscillatory behavior and nonconvergence, mainly owing to the fact that $\mathbf{U}_k = \mathbf{U}$ is only realized when $k = \infty$, as pointed out earlier by Herndon[6] and Davis and Ellzey.[9] The revised iterative procedures such as eqs 14 and 15 accelerate the converging process greatly.[6,9,10] However,

$$\mathbf{U}_k = \mathbf{U}_{k-1} + \mathbf{A}\cdot\mathbf{U}_{k-1} \quad (14)$$

$$\mathbf{U}_k = \mathbf{U}_{k-1} + \mathbf{A}\cdot\sqrt{\mathbf{U}_{k-1}} \quad (15)$$

the arbitrary convergence criterion (eq 8) does not always

Table VI. Vertex Partitionings Induced by U_k's under Different Labelings

iterative procedure	labeling I	n_V	labeling II	n_V	n_Γ
	{1,2,3,4,5,6}		{1,6,7,12,13,18}		
	{7,10,11,14,15,18}		{2,5,11,17}		
eq 14	{8,9,12,13,16,17}	3	{3,4,9,10,15,16}	4	3
			{8,14}		
	{1,2,3,4,5,6}		{1,6,12}		
	{7,10,11,14,15,18}		{2,5,11,14,17}		
eq 15	{8,9,12,13,16,17}	3	{3,4,9,10,15,16}	4	3
			{7,13,18}		

Table VII. Values of n_Γ, n_V, and $|\lambda_2/\lambda_1|$ for Some Triangular Polyacene Graphs

no. of rings	n_V under U_k	n_Γ	$\lvert\lambda_2/\lambda_1\rvert$
4 (18 vertices)	3	3	0.7779
7 (30 vertices)	5	5	0.8809
16 (66 vertices)	16	11	0.9826

guarantee that the generated vector U_k has converged upon the principal eigenvector U, depending on the size and symmetry of the given system. One major problem associated with this arbitrary convergence is that the vector partitioning induced by U_k under the convergence criterion (eq 8) may become very sensitive to the initial labeling of the given molecular graph, especially when the graph under investigation possesses higher symmetry other than the trivial C_1, i.e., onefold axis. Consider the triphenylene molecule as an example shown as graph 3 in Figure 1. The total number of possible labelings for graph 3 is given by $n! = 18!$. Two labelings are randomly chosen to label the molecule, written as permutations:

$$\text{label I}\quad \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 \end{bmatrix}$$

$$\text{label II}\quad \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 \\ 3 & 16 & 15 & 10 & 9 & 4 & 5 & 6 & 7 & 8 & 11 & 12 & 13 & 14 & 17 & 18 & 1 & 2 \end{bmatrix}$$

The vertex partitionings induced by U_k's generated by the module in Table II for the given sequence number assignments are shown in Table VI. For graphs which possess similar symmetries, the problem becomes serious as the number of vertices increases. With no intention to give a rigorous analysis, we speculate that, in cases where the vertex partitioning varies with the initial labeling of the given graph, the generated vector U_k under the convergence criterion (eq 8) could be still at the early stage of the converging process and is thus determined largely by the adjacency matrix, which is not a graph invariant and thus varies with different initial labelings of the given system, despite the fact that the principal eigenvector is theoretically invariant to initial labelings. Another related problem caused by the arbitrary convergence is that for very slow converging processes, which are characterized by the ratio $|\lambda_2/\lambda_1|$ approaching 1 where λ_1 and λ_2 are the largest and the second largest eigenvalues (in magnitude) of the given system, the vector U_k under the convergence criterion (eq 8) often induces vertex partitioning with either $n_V > n_\Gamma$ or $n_V < n_\Gamma$, depending on the size and symmetry of the system under study. In Table VII we list the vertex partitioning under U_k in terms of n_V's for triangular polyacene systems containing 4, 7, and 16 rings, respectively. The corresponding ratios ($|\lambda_2/\lambda_1|$) and n_A's are also given in Table VII. The eigenvalues in Table VII were calculated with the computer code developed on the basis of the Givens–Householder method applied to graphs (see ref 16). It is worth noting that despite the problems discussed above, the vector U_k is generally very stable and discriminating toward the systems possessing only the trivial symmetry C_1, which is especially so when U_k is generated by the iterative procedure in Table VI. Consider graph 4 in Figure 1 as an example. The vertex partitioning induced by U_k is identical with the automorphism partitioning even though the molecule possesses isospectral points:

$$P_\Gamma\text{:}\ \{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\{7\},\{8\},\{9\},\{10\},\{11\},\{12\},\{13\},\{14\}$$

As it should be expected, the vertex partitioning induced by the principal eigenvector U generated by the module in Table IV is stable toward initial labelings but converges slowly. Consider graph 3 in Figure 1 as an example. Table VIII lists the vertex partitioning under U's for the two sequence number assignments given above. The eigenvectors corresponding to the different initial labelings assigned to each vertex with identical 'weights', generated by the module in Table IV, are shown in Table IX. It is easily seen that different labelings simply interchange the rows of the column eigenvector U. Although isospectral points still remain undistinguished, eigenvector U generated by the module in Table IV is stable and discriminating compared with U_k generated by the module in Table III. And as far as eigenvector generating is concerned, the module in Table IV is numerically much more deterministic than the module in Table III. The advantage of the module in Table IV becomes significant when systems of larger sizes and higher symmetry are dealt with.

Consider graph 2 in Figure 1 to demonstrate the use of the last algorithm in Table V. The execution sequences are shown in Table X. Consequently, the vertex partitioning obtained using the new algorithm is

$$V = V_1 \cup V_2 \cup V_3 \cup V_4 \cup V_5 \cup V_6$$

where $V_1 = \{1\}$, $V_2 = \{2\}$, $V_3 = \{4,8\}$, $V_4 = \{5,7\}$, $V_5 = \{6\}$, and $V_6 = \{3\}$. In this case the vertex partitioning generated by the module in Table V is identical with the automorphism partitioning of the graph since each vertex is associated with a list of 'weights' in the Table V module, compared to only one 'weight' in modules shown in Tables II–IV. Hence the vertex-partitioning algorithm described by the module in Table V appears to be quite fast, efficient, and discriminating. The Ellzey–Davis as well as the Herndon algorithm also yields satisfactory results for these graphs.[25]

Table VIII. Vertex Partitionings Induced by U's under Different Labelings

n_V (labeling I)	n_V	(labeling II)	n_V	n_Γ
{1,2,3,4,5,6}		{1,6,7,12,13,18}		
{7,10,11,14,15,18}	3	{2,5,8,11,14,17}	3	3
{8,9,12,13,16,17}		{3,4,9,10,15,16}		

Table IX. Eigenvectors Generated by the Module in Table IV for Two Different Labelings[a]

U under labeling I	U under labeling II
1 0.3446	1(17) 0.1197
2 0.3446	2(18) 0.1833
3 0.3446	3(1) 0.3446
4 0.3446	4(6) 0.3446
5 0.3446	5(7) 0.1833
6 0.3446	6(8) 0.1197
7 0.1833	7(9) 0.1197
8 0.1197	8(10) 0.1833
9 0.1197	9(5) 0.3446
10 0.1833	10(4) 0.3446
11 0.1833	11(11) 0.1833
12 0.1197	12(12) 0.1197
13 0.1197	13(13) 0.1197
14 0.1833	14(14) 0.1833
15 0.1833	15(3) 0.3446
16 0.1197	16(2) 0.3446
17 0.1197	17(15) 0.1833
18 0.1833	18(16) 0.1197

[a] The notation $m(n)$ at the ith row means that the vertex i has sequence number assignment n under labeling I and m under labeling II.

Table X. Execution Sequences for Graph 2 in Figure 1 Using the New Algorithm in Table V

From Step 1	
subset index	vertices
I	1
II	2, 4, 5, 6, 7, 8
III	3

$t' = 3$.

From Step 2		
subset index	vertices	list
I	1	(0,1,0)
II	2	(1,0,1)
	4	(0,1,1)
	5	(0,2,0)
	6	(0,2,0)
	7	(0,2,0)
	8	(0,1,1)
III	3	(0,3,0)

From Step 3	
subset index	vertices
I	1
II	2
III	4, 8
IV	5, 6, 7
V	3

$t = 5$. Since $t \neq t'$, repeat Steps 2 and 3.

From Step 2		
subset index	vertices	list
I	1	(0,1,0,0,0)
II	2	(1,0,0,0,1)
III	4	(0,0,0,1,1)
	8	(0,0,0,1,1)
IV	5	(0,0,1,1,0)
	6	(0,0,0,2,0)
	7	(0,0,1,1,0)
V	3	(0,1,2,0,0)

From Step 3	
subset index	vertices
I	1
II	2
III	4, 8
IV	5,7
V	6
VI	3

$t = 6$. Since $t \neq t'$, repeat Steps 2 and 3.

From Step 2		
subset index	vertices	list
I	1	(0,1,0,0,0,0)
II	2	(1,0,0,0,0,1)
III	4	(0,0,0,1,0,1)
	8	(0,0,0,1,0,1)
IV	5	(0,0,1,0,1,0)
	7	(0,0,1,0,1,0)
V	6	(0,0,0,2,0,0)
VI	3	(0,1,1,0,0,0)

From Step 3	
subset index	vertices
I	1
II	2
III	4, 8
IV	5, 7
V	6
VI	3

$t = 6$. Since $t = t'$, the module is terminated after three iterations.

As examples of transitive graphs and regular graphs, we considered the graphs shown in Figure 2 and numerous other graphs such as the recently synthesized buckminsterfullerene[27] or soccerballene (C_{60}), archimedene (C_{120}) discussed in ref 28, and a hand form of C_{60} discussed in ref 16. For all these graphs, the ring-partitioning method followed by the vertex-

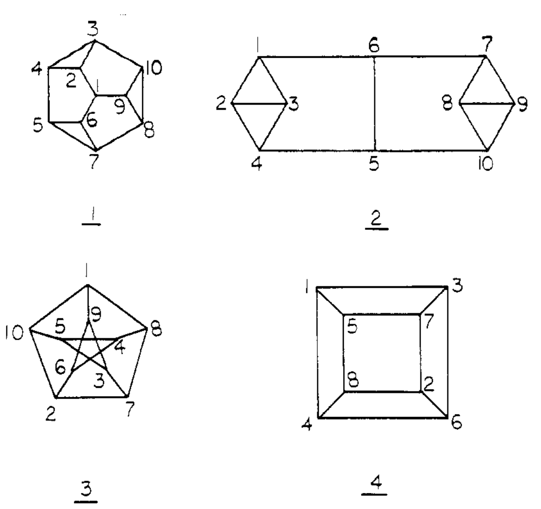

Figure 2. Four regular graphs of chemical interest. Note that two of these are vertex-transitive graphs.

partitioning technique was invoked. For graphs 1 and 2 in Figure 2, the partitioned vertex sets are obtained as

graph 1

$$V = \{1\} \cup \{2,6,9\} \cup \{3,4,5,7,8,10\}$$

graph 2

$$V = \{1,4,7,10\} \cup \{2,3,8,9\} \cup \{5,6\}$$

For the Peterson graph, cubane graph, Soccerballene graph (C_{60}), and archimedene graph (C_{120}), we obtained a single vertex set. Although the final outcomes for these graphs are trivial since these are transitive graphs, the fact that the newly developed algorithms provided satisfactory solutions to these graphs is worth noting since these graphs could be considered "pathological" due to their transitive nature.

CONCLUSION

Considering that only the vertex partitioning, which is invariant under automorphism, leads to the automorphism partitioning, a new vertex-classifying algorithm was developed. This was compared with many existing algorithms. It was found that the new algorithm is superior to the Morgan algorithm as well as the principal eigenvector algorithm. Although difficult to implement, the vertex-partitioning algorithm readily distinguishes isospectral points, is not interfered with different initial labelings and oscillatory behavior, and most importantly, always converges rapidly. On the other hand, when the principal eigenvector scheme is used to induce partitioning among vertices, the algorithm should be the top choice for generating the eigenvector U. Although several algorithms are not completely sufficient to generate the automorphism partitioning among vertices for all molecular graphs, especially highly transitive graphs, the induced vertex partitioning serves as a starting point for further searching of the automorphism partitioning. However, for completely transitive graphs the solution is trivial in that there is only one equivalence class of vertices by definition.

ACKNOWLEDGMENT

This research was supported by the National Institutes of Health Grant GM37963. We thank Dr. M. Razinger for many useful comments on this manuscript which improved the presentation significantly.

REFERENCES AND NOTES

(1) Morgan, H. L. *J. Chem. Doc.* **1965**, *5*, 107.
(2) Wipke, W. T.; Dyott, T. M. *J. Am. Chem. Soc.* **1974**, *96*, 4836.
(3) Moreau, G. *Nouv. J. Chim.* **1980**, *4*, 17.

(4) Golender, V. E.; Drboglav, V. V.; Rosenblit, A. B. *J. Chem. Inf. Comput. Sci.* **1981**, *21*, 196.
(5) Razinger, M. *Theor. Chim. Acta* **1982**, *61*, 581.
(6) Herndon, W. C. *Inorg. Chem.* **1983**, *22*, 554.
(7) Herndon, W. C. *Tetrahedron Lett.* **1974**, 671.
(8) Herndon, W. C. *J. Chem. Doc.* **1974**, *14*, 150.
(9) Davis, M. I.; Ellzey, M. L., Jr. *J. Comput. Chem.* **1983**, *4*, 267.
(10) (a) Randić, M.; Davis, M. I. *Int. J. Quantum Chem.* **1984**, *24*, 69. (b) Randić, M.; Brissey, G. M.; Wilkins, C. W. *J. Chem. Inf. Comput. Sci.* **1981**, *21*, 52. (c) Randić, M. *J. Chem. Inf. Comput. Sci.* **1977**, *17*, 171.
(11) Shelley, C. A.; Munk, M. E. *J. Chem. Inf. Comput. Sci.* **1979**, *19*, 171.
(12) (a) Shelley, C. A.; Munk, M. E. *J. Chem. Inf. Comput. Sci.* **1979**, *19*, 247. (b) Munk, M. E.; Christie, B. D. *Anal. Chem. Acta*, **1989**, *212*, 57.
(13) Gibbons, A. *Algorithmic Graph Theory*; Cambridge University: London, 1985.
(14) Wilkinson, J. H. *The Algebraic Eigenvalue Problem*; Clarendon, Oxford, 1965.
(15) Jennings, A. *Matrix Computation for Engineers and Scientists*; John Wiley & Sons: New York, 1977.
(16) Balasubramanian, K.; Liu, Xiaoyu *J. Comput. Chem.* **1988**, *9*, 406.
(17) Householder, A. S. *The Theory of Matrices in Numerical Analysis*; Blaisdell: New York, 1964.
(18) Householder, A. S.; Bauer, F. L. *Numer. Math.* **1959**, *1*, 29.
(19) Wilkinson, J. H. *Comput. J.* **1960**, *3*, 23.
(20) Ortega, J. In *Mathematical Methods for Digital Computers*; Ralson, Antony, Wilkf, Herbert S., Eds.; Wiley: New York, 1967; Vol. II, pp 65-93.
(21) Corneil, D. G.; Gotlieb, C. C. *J. Assoc. Comput. Mach.* **1970**, *17*, 51.
(22) Read, R. C.; Corneil, D. G. *J. Graph Theory* **1977**, *1*, 339.
(23) Herndon, W. C.; Ellzey, M. L., Jr. *Tetrahedron* **1975**, *31*, 99.
(24) Liu, X. Y.; Balasubramanian, K.; Munk, M. E. *J. Magn. Reson.* **1990**, *87*, 547.
(25) Herndon, W. C. In *Chemical Applications of Topology & Graph Theory*; King, R. B., Ed.; Elsevier: Amsterdam, 1983.
(26) Wipke, W. T.; Dyott, T. M. *J. Chem. Inf. Comput. Sci.* **1975**, *15*, 140.
(27) Heath, J. R.; O'Brien, S. C.; Zhang, Q.; Liu, Y.; Curl, R. F.; Kroto, H. W.; Tittle, F. K.; Smalley, R. E. *J. Am. Chem. Soc.* **1985**, *107*, 7779.
(28) Balasubramanian, K.; Liu, X. Y. *Int. J. Quantum Chem. Symp.* **1988**, *22*, 319.

Simple and Fast Search System for Closely Related Proteins

SHIN-ICHI NAKAYAMA,* KATSUKO SUGAI, YURIKO HOTATE, and MASAYUKI YOSHIDA

University of Library and Information Science, Tsukuba-city, Ibaraki, 305 Japan

Received March 14, 1990

A method for measuring a probability distance from the Euclidean distance between proteins, which were expressed by use of all possible paris of amino acids as descriptors, is described. A system to search for closely related proteins by the probability distance between proteins was constructed, and its advantage over other systems is discussed.

INTRODUCTION

Some properties of a little-studied protein would be predicted from the knowledge of well-characterized proteins of similar amino acid sequences. Likewise, evolutionary relationships among biological species would be revealed by sequential similarities between proteins. Thus, if an amino acid sequence of a new protein were revealed, it is necessary that one should search for proteins with similar sequences.

Meanwhile, sequence data of many proteins have been accumulated, and various methods to search for similar proteins from sequence data have been developed. In most of them sequential similarities were measured on the basis of the maximum match between amino acid sequences.[1] This process is, however, very time-consuming. It takes 8 h to search for a protein of 200 residues with 2600 proteins (about 500 000 residues) in the NBRF protein library by a computer program SEQHP on the VAX11/750 computer. A recently developed program FASTP greatly cut short the execution time to 2-5 min under the same condition.[2] However, the number of proteins included to the NBRF database has increased steadily to 5251 (1 384 621 residues) in March 1988, and development of a faster search method is now required.

Previously, we have presented a method for clustering proteins using all possible pairs of amino acids as descriptors.[3] This paper describes the simple and fast search system for proteins based on an improved similarity measurement.

A METHOD FOR MEASURING PROBABILITY DISTANCE

All possible pairs of amino acids, which numbered 400, were assigned as the descriptors that reflect one-dimensional structures of proteins. Protein i (P_i) was then expressed by a set of descriptor values as $P_i = (x_{i1}, x_{i2}, ..., x_{ik}, ..., x_{i400})$, where x_{ik} was the frequency of occurrence of the kth descriptor in protein i and was readily derived from its amino acid sequence.

The Euclidean distance d_{ij} between proteins i and j was then calculated by eq 1. The d_{ij} value is just a number, and how

$$d_{ij} = \sqrt{\sum_{k=0}^{400} (x_{ik} - x_{jk})^2} \qquad (1)$$

close the distance is between proteins i and j is not known from its magnitude. Ideally, one would prefer to know the exact probability of obtaining a given d_{ij} value, but that is prohibitively difficult.

As an alternative to determining the exact probability, it is possible to estimate it experimentally. Thus, a pair of random amino acid sequences i and j (A_i and A_j) having the same lengths as proteins i and j was produced, and the Euclidean distance ad_{ij} for that pair was calculated. In a way similar to the above the distance ad_{ij} was calculated for 200 pairs of A_i and A_j produced independently. A plot of the ad_{ij} values thus obtained against the frequency of the pairs showed a normal distribution, from which a mean distance m_{ij} and a standard deviation σ_{ij} were calculated. The division of the difference between d_{ij} and m_{ij} by σ_{ij} gives the probability distance Z_{ij} between proteins i and j (eq 2).

$$Z_{ij} = (d_{ij} - m_{ij})/\sigma_{ij} \qquad (2)$$

EMPIRICAL EQUATIONS FOR THE MEAN DISTANCE AND THE STANDARD DEVIATION

In the calculations of Z_{ij}, estimation of m_{ij} and σ_{ij} takes a long time. Thus, empirical equations for them were constructed.

The data sets 1, 2, and 3 of m_{ij} and σ_{ij} were produced experimentally for type 1 pairs of A_i and A_j with shorter