Learning Approach for the Computation of Generalized Wheland Polynomials of Chemical Graphs

M. M. Balakrishnarajan and P. Venuvanalingam*

Department of Chemistry, Bharathidasan University, Tiruchirappalli 620 024, Tamil Nadu, India

Received March 1, 1994®

The generalized Wheland polynomial of a graph is computed as perfect matchings of the associated edge-weighted complete graph. A depth-first-search algorithm with unit penetrance is developed for an effective combinatorial exploration of all perfect matchings of the edge-weighted complete graph, using *a priori* control knowledge. Since this is a typical *NP*-class problem, dynamic learning techniques are designed to push the tractable limit of the problem further. The choice of the learning technique depends solely on the amount of memory available for use since demand for memory increases concomitantly with knowledge acquisition. The skeleton of the algorithm is presented in a Pascal-like *pseudo* language.

1. INTRODUCTION

Generalized Wheland polynomial is a chemical graph-theoretical invariant that enumerates the valence structures of different degrees of excitation for conjugated hydrocarbons i.e., the kth term in the polynomial corresponds to valence structures in which $2(k-1)$ nonadjacent vertices are coupled. The edges connecting the nonadjacent vertices are called long bonds or edges and a valence structure with kth degrees of excitation will have k numbers of long bonds. Here a valence structure refers to a graph represented by an edge set. Correspondingly, a valence structure with kth degrees of excitation represents the parent graph with k numbers of (nonexistent) long bonds. The generalized Wheland polynomial of a graph with n vertices is defined[1] as

$$\mathrm{Wh}_G(x) = \sum_{k=0}^{n/2} w(G;k)x^k \qquad (1.1)$$

where $w(G;k)$ counts the valence structures of kth degrees of excitation. $w(G;k)$ is closely related to $p(G;k)$, which counts the different ways of selecting k numbers of disjoint edges on G. In fact $w(G;0)$ is the Kekule structure count of G that is identical to $p(G;n/2)$.

The generalized Wheland polynomial has been proved to be a structural invariant, though not unique, for a given graph. The generalized Wheland polynomial has been defined by Randić and co-workers,[1] and it is a modification of the Wheland polynomial that was originally defined by Wheland.[2] This original polynomial depends upon the choice of labeling and hence is not a structural invariant. It was defined for deriving the number of valence structures in different degrees of excitation that was used with valence bond (VB) computations. VB computation requires a set of valence structures that are called canonical structures. Initially, valence structures with overlapping bonds were considered "noncanonical" and were omitted in VB computations and later, these structures were also included in detailed valence bond computations. Randić and co-workers have defined generalized Wheland polynomial involving such numerous additional valence structures that were ignored in the Wheland polynomial.[2-4] Apart from VB computations, the generalized Wheland polynomial is particularly useful in discriminating graphs having identical Kekule structure count as found in the case of chrysene and benzophenanthrene (Table 1).

There are only a few reports[1,5,6] on the generalized Wheland polynomial and so far only one algorithm[6] to compute this. Hand computation of this polynomial becomes unwieldy even for graphs with 10 vertices, and the number of canonical structures increases explosively with the number of vertices of the graph. Because of such combinatorial explosion, the problem becomes intractable even for graphs of moderate size. It has been reported that the existing program needs 3 days and 19 h of computation time for a graph with 18 vertices. Computation of this polynomial is an *NP* class problem (nondeterministic polynomial), like that of matching polynomial and chromatic polynomial, and no traditional methods could be applied to compute this. NP class problems are defined as problems that can be solved in polynomial time by a nondeterministic Turning machine (NDTM),[7] and their time complexity is exponential. In such circumstances, as we found earlier,[8,9] search-based enumeration with heuristic enhancements could be employed to compute this polynomial. In the present work, we report a novel enumerative algorithm that is capable of dynamic learning to compute the generalized Wheland polynomial. It should be noted that the earlier algorithm[6] has several limitations; most importantly, it lacks consistency with respect to graph size and its requires an unconventional input scheme.

We have developed a weighted perfect matching model for counting resonance structures of various degrees of excitation, and thus computation of the generalized Wheland polynomial is reduced to the perfect matching enumeration of the associated edge-weighted complete graph. Such enumeration involves symbolic computation as opposed to number crunching, which demands search-based enumerative schemes. By symbolic computation, we mean manipulation of edges as symbols rather than numbers or aggregates of numbers. Due to the NP nature of this problem, we use heuristics and other artificial intelligence (AI) techniques like learning to push the limit of combinatorial explosion further. In effect, we develop a consistent and compact algorithm for the rapid computation of generalized Wheland polynomials of graphs.

2. OUTLINE OF THE MODEL

For the purpose of computing $w(G;k)$ of G with n vertices, we define a complete graph K_n that has two type of edges: edges with weight 1, corresponding to edges present in the given graph G, and those with zero, nonexisting edges in the

* Author to whom correspondence should be addressed.
® Abstract published in *Advance ACS Abstracts*, July 1, 1994.

0095-2338/94/1634-1113$04.50/0

Table 1. Generalized Wheland Polynomials of Graphs Given in Figure 3

N	G	generalized Wheland polynomial
18	a	$9 + 117x + 2124x^2 + 21375x^3 + 156195x^4 + 824751x^5 + 3122613x^6 + 8077176x^7 + 12821220x^8 + 9433845x^9$
18	b	$8 + 116x + 2090x^2 + 21020x^3 + 155651x^4 + 824732x^5 + 3126164x^6 + 8078180x^7 + 12813740x^8 + 9437724x^9$
18	c	$8 + 117x + 2100x^2 + 21087x^3 + 155613x^4 + 824442x^5 + 3125958x^6 + 8079675x^7 + 12812238x^8 + 9438187x^9$
18	d	$4 + 219x + 3774x^2 + 39339x^5 + 259734x^4 + 1220736x^5 + 4020666x^6 + 8962221x^7 + 12216306x^8 + 7336426x^9$
20	e	$9 + 220x + 4422x^2 + 55254x^3 + 505281x^4 + 3411765x^5 + 17101350x^6 + 62217888x^7 + 155991651x^8 + 241496124x^9 + 173945111x^{10}$
20	f	$36 + 1040x + 19800x^2 + 209340x^3 + 1539825x^4 + 8243928x^5 + 32392710x^6 + 91908900x^7 + 179525160x^8 + 217342800x^9 + 123545536x^{10}$

same. Thus, it is an edge-weighted complete graph associated with the given graph G. The weights assigned as 1 and 0 for normal and long bonds are arbitrary. Thus the edge-weighted adjacency matrix is defined as

$$w(i,j) = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are adjacent in the parent graph} \\ 0 & \text{otherwise} \end{cases}$$

and is identical to the adjacency matrix of G. In other words any graph G of n vertices can be represented as a weighted complete graph K_n and the edge set $E(G)$ of G is a subset of the edge set $E(K)$ of K_n. The members of subset $E(G)$ will be assigned arbitrarily a weight 1, and the remaining members of $E(K)$ are assigned a zero weight. The edges with zero weight are nonexistent in the given graph G and are considered long bonds of excitation, during the enumeration of $w(G;k)$. Given that the enumeration is based on K_n, one has to handle $n(n-1)/2$ edges irrespective of the actual number of edges in $E(G)$. It is because of this reason that the problem becomes intractable even for small values of n like 16. For instance, the number of edges to be handled for graphs with 16 and 18 vertices are respectively 120 and 153. It will be shown later that combinatorial explosions definitely occur in these ranges, and even if powerful heuristics are used, the limits could not be pushed much further.

Once such an edge-weighted complete graph K_n is defined, the coefficients of the generalized Wheland polynomial $w(G;k)$ are enumerated as perfect matchings of K_n. Thus the computation of the generalized Wheland polynomial has been converted to perfect matching enumeration as already mentioned.

In this context it is worthwhile to review the earlier work done on such enumerations. Though much work has been done on Kekule structure enumeration, there is no single algorithm in the chemical literature that can be generalized to enumerate the perfect matchings of edge-weighted complete graphs. A detailed discussion on the existing algorithms on perfect matchings on graphs has been done recently.[9] In mathematical literature, algorithms for generating a single edge-weighted maximal matching are available.[10,11] But, none of them is adaptable for the present problem as it needs each and every perfect matching of the given graph. Therefore, we design an efficient algorithm based on search to compute the generalized Wheland polynomial of a graph. The efficiency of the present method is shown in the final section.

3. ALGORITHM DESCRIPTION

We have already described that search is the best technique for the computation of perfect matchings on graphs.[9] As the problem of computing the generalized Wheland polynomial is now reduced to perfect matching enumeration, the present algorithm is also designed on enumerative search. Though the nature of the problem falls well within the scope of artificial intelligence, it does not require the sheer bulk of a knowledge base or numerous inference rules. Our prime concern is speed without using approximate heuristic estimates. Hence, the obvious choice for coding is any high-level language that is capable of recursion. The algorithm discussed here is given in the Appendix.

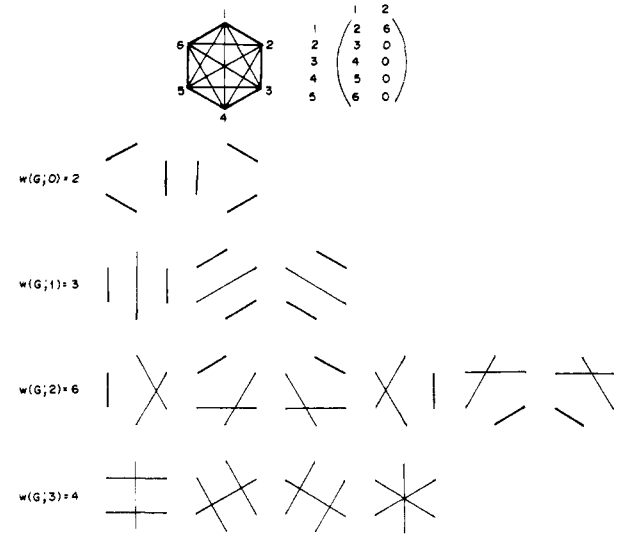

Figure 1. Weighted graph, compact matrix, and all possible perfect matchings of the benzene graph.

The choice of abstract data type for input is made in a way that facilitates easier generation of matchings. Since the weighted and adjacency matrix is very sparse and requires n^2 input values, a compact matrix $\mathbf{C}[i,j]$ comprising the same adjacency information of the labeled graph is defined. Each nonzero entry $\mathbf{C}[i,j]$ represents that (1) vertex i is connected to $\mathbf{C}[i,j]$, (2) $\mathbf{C}[i,j]$ is greater than i, and (3) $\mathbf{C}[i,j]$ is the jth smallest vertex connected to i.

For a graph $G(V,E)$ with n vertices, the row values range from 1 to n and the column values from 1 to the maximal degree of G. From this information the weighted adjacency matrix is generated. As an illustration, the edge-weighted benzene graph, its compact matrix, and its various degrees of excitation are given in Figure 1.

Though the above described information is sufficient, the symbols upon which the algorithm is operating are the list of all of the edges of the complete graph. All symbols of a complete graph can be listed as two-combinations of n vertices, i.e., $\binom{n}{2}$. But, for avoiding duplication of symbols, the graph-theoretical model chosen is slightly modified by adding a directional nature to the edges. The direction of the edge is arbitrarily fixed so that the direction is always from a lower labeled vertex to the higher one. Implementation of the above constraint removes the ambiguity of listing the symbols $\langle x,y \rangle$ and $\langle y,x \rangle$ as different edges. The group of edges having the same starting vertex is indexed and sorted to form a fine structure of the edge list. The explicit storage of connectivity information is avoided in the present case as the dynamic generation of the edge list can be done cost effectively. Two nested loops with variables Var1 and Var2, of whom Var1 ranges from 1 to $n-1$ and Var2 from var1 to n, can be conveniently used as an edge generator to form all of the two-combinations of symbols.

Problem Representation. In artificial intelligence, the representation problem is the transformation of the problem

statement into the components of a production system. The production system comprises three vital parts, namely, the global database, the rules of inference, and the control strategy.

First and foremost is the global database that represents the information about the nodes of the search tree. This consists of matchings, their weights, and other relevant information for algorithmic enhancements that will be discussed in detail later. The disjoint edge set in the matching is represented in terms of vertices using a Boolean array. Though the exact adjacency information of the matching is lost by choosing this data structure, it is found to be effective when one is interested only in coefficients of the polynomial. The weights of the nodes will be another data item in the node record. The final results, i.e., coefficients of the generalized Wheland polynomial, are represented by a one-dimensional array, whose elements represent the number of perfect matchings with varying weights.

Using this infrastructure, the disjoint-set-union rule is framed to deduce various expressions corresponding to the nodes of the search tree. The disjoint-set-union rule checks if the matching of the given node and the selected edge shares a common vertex. If yes, the edge will be ignored. Otherwise, the successor node is generated by inserting the edge vertices in the Boolean array of the current node, and the weight of the successor is updated using the weighted adjacency matrix. All remaining edges are tested with the current node to generate all possible successors. This is repeated over the entire edge set $E(k)$. For a graph with n vertices the number of nodes $M(n)$ that has to be generated is given by the recursive equation

$$M(n) = \sum_{i=1}^{n-1} i(M(i-1)+1) \qquad (3.1)$$

where $M(0) = 0$. As mentioned in the earlier section, edges are grouped so that they share a common vertex referred to as the group index. Naturally an edge in a group cannot have a disjoint partner within the same group. Therefore, while looking for disjoint edges, searching within the group can be avoided. Also, the effective number of decision steps can be reduced by ignoring the entire group belonging to the nondisjoint index. This reduces the effort needed to generate the search tree. The complete tree can be grown using this single rule. The mathematical basis of this implication is straightforward.[9]

The obvious choice of control strategy for inference to the structure described above is depth first in order to minimize the memory requirements because of the rapid widthwise growth of the tree. A backtracking control regime serves the purpose as it merely keeps track of the path that it is working on currently, modifying it when necessary. This avoids direct proportioning of the memory with respect to tree width. This is implemented using recursion.

Thus a stage is set for search. If the search is done "uninformed", i.e., by brute force, a lot of effort will be wasted in blind alleys, paths that are not leading to perfect matching. With the increase in graph size, the number of such blind alleys will increase explosively. Thus, the AI techniques that are employed during search result in a substantial time gain and hence push further the limit of combinatorial explosion that will otherwise set off very early. We describe the AI techniques we use in the following sections.

4. HEURISTIC PERCEPTION OF BLIND ALLEYS

The production of matching into perfect matching entails a series of monotonic, but not blind, disjoint-set-union paths. In the matching tree[8] only certain paths lead to perfect matching, and while enumerating perfect matching, it is sufficient to traverse these paths alone. Other paths that do not lead to perfect matchings are blind alleys, and they can be avoided using heuristic guidance. *A priori* knowledge of blind alleys is acquired by the use of the following theorem.

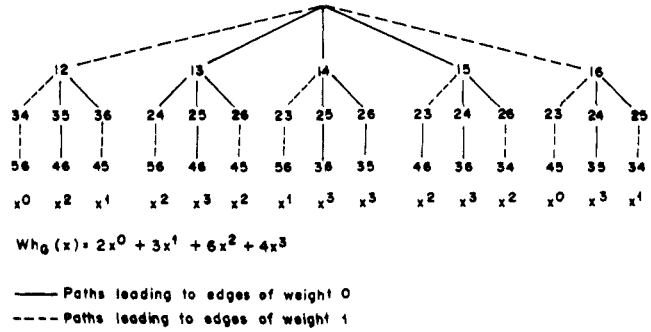

Figure 2. Search tree of benzene that is generated and traversed by the inference mechanism using heuristics.

Theorem 1. The edges of the first found index group of the edge list alone lead to the necessary and sufficient number of paths that are to be traversed.

Let j be the index disjoint to the matching under consideration. Since the possibility of finding vertex j in the groups trailing after the jth group is nil, it is not possible to form any perfect matching without a member from the jth group. Hence, it is compulsory to select an edge from the jth row. The following corollary can be deduced using theorem 1.

Corollary. Every Disjoint edge $\langle j,i \rangle$ in the group of the first found index j leads to at least one perfect matching.

As we are dealing with complete graphs, any two combination of n vertices will be an edge. Consequently, the vertices absent in the current matching can be obtained by at least one set of two combinations. Hence, it is sufficient if we restrict admissible paths by expanding every node using the edges of the first disjoint group. The rest of the successors are blind alleys.

The heuristic implementation of the above facts forces the inference mechanism to select only the paths that lead to perfect matching and thereby scissors all blind alleys in packs. In addition, the ground existence of these heuristics poses no new problems for computation and adds *a priori* knowledge to the control strategy which is independent of the graph that is explored.

Efficiency of the heuristic function is measured in terms of penetrance and accuracy. Penetrance is the ratio between the minimum distance between the start and goal states and the number of nodes actually traversed. Since all the disjoint edges in the first found index group lead to perfect matching, the penetrance of the search achieves an ideal value of unity. For an n vertex graph, the number of nodes $H(n)$ that has to be traversed is now reduced as

$$H(n) = \sum_{i=1}^{n/2} \prod_{j=1}^{i} n - 2j + 1 \qquad (4.1)$$

The heuristic tree for benzene is given in Figure 2. Due to the theorematic accuracy of the heuristic estimation, the control moves straight toward the goal states, giving accurate results. Nonetheless, the heuristic limitation of successors does not change the exponential nature of the algorithm. Only the explosion limit of combinatorial growth is pushed further. It is found that a graph of 18 vertices requires around 18 h of computation without heuristics and 25 min when heuristics are applied. The above observation shows well the exact impact of these heuristics on the inference mechanism.

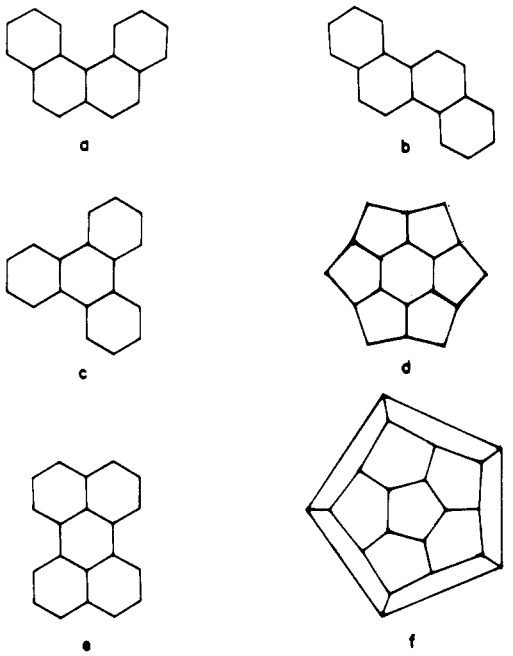

Figure 3. Graphs of certain important molecules as listed in Table 1.

The algorithm described above can be implemented even in a personal computer, and results for graphs containing 20 or fewer vertices can be obtained, as it requires very little memory. The Pascal program developed on the basis of our algorithm has taken only 11 min for graphs with 18 vertices in a PC 4AT6 SX computer with a 25-MHz clock speed. The same input size requires 3 days and 19 h by the earlier algorithm on a VAX 11/370 system. For 20 vertex graphs our program requires 3 h and 23 min of CPU time. Generalized Wheland polynomials for certain novel chemical graphs (Figure 3) are reported for the first time (Table 1). For graphs of smaller sizes, i.e., $n < 16$, the program takes only a few seconds.

Though heuristic scissoring of blind alleys improves the efficiency of the search process, the tree grows explosively even with the attainment of unit penetrancy. Computation of the generalized Wheland polynomial of larger graphs using the heuristic searching discussed above demands more time. In such cases, searching requires much more intelligence, and it is found that learning, a very powerful AI technique, can be adopted in this case. Searching the nature of the algorithm forms the skeleton to which these skills can be embedded with relative ease.

5. LEARNING AND KNOWLEDGE ACQUISITION

One of the most desirable features of an intelligent system is its ability to learn and acquire knowledge and use it during subsequent execution. Identifying the similarities of the instances and recalling past experience and knowledge are critical processes of learning systems. Since blind alleys are completely removed from the search domain by heuristics, improvements of the system can only be done using learning.

Learning here involves gathering new knowledge from problem solving experiences which were previously unavailable to the inference mechanism. Learning can be used here because there are identical nodes in the search tree, whose successor nodes will require the same edge sequence for production. For example, the node ⟨1,2,3,4⟩ formed by the disjoint-set-union of edge ⟨1,2⟩ and ⟨3,4⟩ is equivalent to the nodes formed by edges ⟨1,3⟩, ⟨2,4⟩ and ⟨1,4⟩, ⟨2,3⟩. Likewise, identical nodes containing three or more edges can also be found in different parts of the search tree. Situations like this

Table 2. Efficiency Improvements in Enumerative Searching due to Heuristics and Learning[a]

N	$G(n)$	$M(n)$	$H(n)$	$L(n)$	$I(n)$
2	1	1	1	1	1
4	3	9	6	7	4
6	15	75	35	31	12
8	105	763	252	127	33
10	945	9 495	2 277	511	88
12	10 395	140 151	25 058	2 047	232
14	135 135	2 390 479	325 767	8 191	774
16	2 027 025	46 206 735	4 886 520	32 767	1 596
18	34 459 425	997 313 823	83 070 857	131 071	3 976
20	654 729 075	23758 664 095	1578 346 302	524 287	10 945

[a] Nodes: $G(n)$, no. of goal nodes; $M(n)$, no. of matching tree nodes; $H(n)$, no. of heuristic tree nodes; $L(n)$, no. of learning tree nodes; $I(n)$, no. of heuristic and learning tree nodes.

are suitable for learning, since the first occurrence of such a node can be used to gain experience and the nature of the perfect matchings obtained can be stored as knowledge bases. During the instance of multiple occurrences of such identical nodes, the knowledge stored can be recalled and used in computations without actually traversing the branches beyond the current depth. It is found that learning can play a vital role in this problem since the perfect matching tree is well characterized.

The first step of the learning process is the identification of identical nodes. This requires the information of the nodes that was previously learned during search. In the final step the previous result (weights of perfect matchings) is recalled and modified so as to compensate for the difference in weights of the perfect matchings. This type of learning requires data caching of certain branches that form nonidentical nodes. This process requires enormous amounts of memory when the system is designed to learn and gather all its experience during traversal. Precisely, for a graph G with n vertices the number of distinct nodes of the tree is given by

$$L(n) = \sum_{k=1}^{n/2} \binom{n}{2k} \tag{5.1}$$

It should be noted that all of the possible combinations of even length form a distinct node of the tree and form a part of the knowledge base. In addition to this, every nonidentical node is associated with an array that contains the weights of the branches that leads to perfect matchings. Interestingly, it is found that heuristic enhancements discussed in the earlier section can be intertwined along with the learning attribute without any additional computational cost. The resulting inference mechanism is found to be very powerful and can be easily verified from Table 2. Here, the number of nodes that have to be traversed by the inference mechanism by *brute force* and by using heuristics for various graph sizes is given. The impact of learning in limiting the number of nodes of brute force search is given in the fourth column. An ideal intelligent search in which both learning and heuristics are intertwined inextricably will result in a dramatic reduction of the number of nodes that is given by

$$T(n) = \sum_{i=1}^{n/2} \binom{n-i}{i} \tag{5.2}$$

The last column of Table 2 maps the number of nodes for this intertwined search.

Though learning exerts a considerable effect on the time complexity of search, unlike the heuristic technique discussed earlier, learning demands an increasing amount of

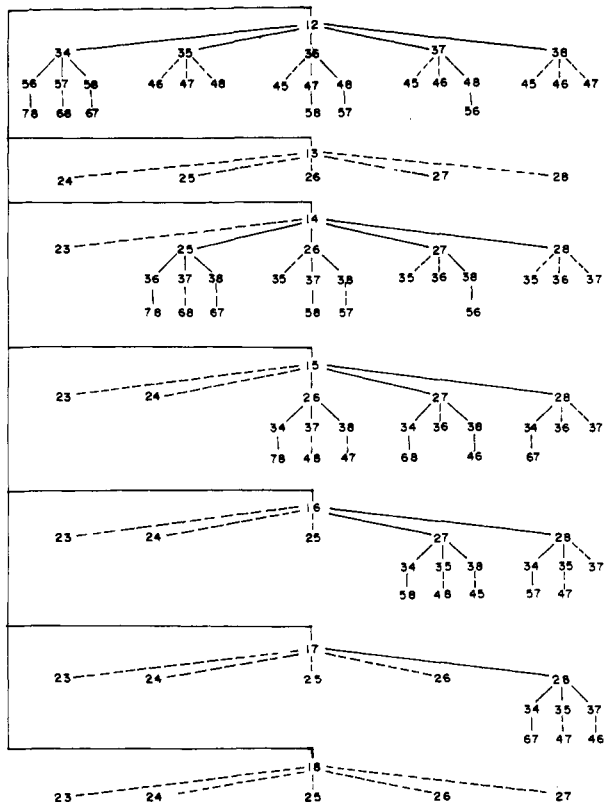

Figure 4. Combined heuristics and learning tree for any graph with eight vertices. Sharp lines denote paths traversed and broken lines denote paths avoided due to learning.

memory. For a normal control strategy, the memory requirements for learning increases exponentially. Hence, the control mechanism has to be built so as to optimize the number of nodes that are dynamically preserved during execution. The feasible way is to reduce the limits of learning according to the availability of memory resources. Different learning strategies with varying degrees of learning abilities can be adopted accordingly.

The degree of learning of this system can be measured in terms of the number of edge sequences it learns, to acquire knowledge. The second degree learning described earlier in the case of ⟨1,2,3,4⟩ is illustrated in Figure 4. Here every consequent disjoint pair of edges in the search tree is preserved along with the information regarding the weight of the branches descending from the present node. During subsequent execution, every disjoint successor must be checked with previously stored knowledge to check vertex similarity. If any of the nodes traversed earlier has the same set of nodes with the present node, then the knowledge stored in that node will be recalled. Then, the control backtracks to the parent node instead of exploring further depths. The branches neglected due to learning are shown as dotted lines in Figure 4.

Apart from these methods, various other schemes such as identification of isomorphic edges in the complete graph through a procedure can be used along with this skeleton, but unlike learning, they are efficient only for selective domains.[12]

6. CONCLUSION

A search-based enumerative algorithm is designed for computing the generalized Wheland polynomials of graphs. For the purpose of computing this polynomial of the graph G, an associated edge-weighted complete graph is defined and coefficients of the generalized Wheland polynomial are computed as perfect matchings of the above complete graph. Enumeration is prohibitively difficult even for graphs of moderate size due to the nondeterministic polynomial character of the problem on the one hand and involvement of complete graphs on the other. Therefore, the dynamic learning techniques and suitable heuristics are derived to enhance the tractable limit of the problem. The algorithm is found to be very efficient.

ACKNOWLEDGMENT

We thank Prof. K. Balasubramanian, Arizona State University, for providing us his reprints. P.V. thanks the Indian National Science Academy for awarding a visiting fellowship for the year 1991–2 during which period this work was conceived. We thank the reviewers for their useful comments.

APPENDIX

The algorithmic representation of the search skeleton is given in *pseudo* for the generation of all perfect matchings on a general graph. The matching of the nodes can be conveniently represented by a Boolean array or by a set type of abstract data that performs bit manipulation.

```
Procedure WHELAND(    IDX  : Index of the current edge group
                  VAR NODE : BOOLEAN ARRAY of matching )

 REPEAT
   BEGIN
     IF NODE [IDX] = FALSE THEN
      BEGIN
          FOR ALL edges <IDX,Y> such that ( IDX < Y < N ) DO
            BEGIN
               IF NODE [Y] = FALSE THEN
                 BEGIN
                  Add edge <IDX,Y> to NODE
                  NODE-WEIGHT := NODE-WEIGHT + WEIGHT <IDX,Y>
                  IF NODE is not perfect matching THEN
                    WHELAND ( IDX, NODE )
                    ELSE RESULT [ WEIGHT] := RESULT [NODE-WEIGHT] +1
                  Remove <IDX,Y> from NODE
                 ENDIF
              ENDFOR
            ENDFOR
      ENDIF
      IDX <-- IDX + 1
   UNTIL NODE [IDX] is false or (IDX > N)
 END
```

REFERENCES AND NOTES

(1) Randić, M.; Hosoya, H.; Ohkami, N.; Trinajstić, N. The Generalized Whelan Polynomial. *J. Math. Chem.* **1987**, *1*, 97–122.
(2) Wheland, G. W. The number of canonical structures of each degree of excitation for and unsaturated or aromatic hydrocarbons. *J. Chem. Phys.* **1935**, *3*, 326–35.
(3) Knop, J. V.; Trinajstić, N. Chemical Graph Theory. II. On the Graph Theoretical Polynomials of Conjugated Structures. *Int. J. Quantum Chem., Quantum Chem. Symp.* **1986**, *20*, 504–20.
(4) Okhami, N.; Hosoya, H. Wheland polynomial I. Graph theoretical analysis of the contribution of the excited resonance structures to the ground state of acyclic polymers. *Bull. Chem. Soc. Jpn.* **1979**, *52*, 1624–33.
(5) Trinajstić, N.; Klein, D. J.; Randić, M. On Some Solved and Unsolved Problems of Chemical Graph Theory. *Int. J. Quantum Chem., Quantum Chem. Symp.* **1986**, *20*, 699–742.
(6) Kopecky, K. J.; Randić, M. Computer generation of generalized Wheland polynomials. *Comput. Chem.* **1987**, *11*, 29–40.
(7) Aho, A. V.; Hopcroft, J. E.; Ullman, J. D. *Data structures and algorithms*; Addison Wesley: New York, 1985.
(8) Balakrishnarajan, M. M.; Venuvanalingam, P. A General Method for the Computation of Matching Polynomials of Graphs. *J. Math. Chem.*, in press.
(9) Balakrishnarajan, M. M.; Venuvanalingam, P. An Artificial Intelligence Approach for the Generation and Enumeration of Perfect. Matching on Graphs. *Comput. Math. Appl.*, in press.
(10) Kuhn, H. W. Variants of the Hungarian method for assignment problems. *Nav. Res. Logist. Q.* **1956**, *3*, 253–8.
(11) Edmonds, J.; Johnson, E. Matching: A well-solved class of integer linear programs. *Combinatorial structures and their applications*; Gordon & Breach: New York, 1970; pp 89–92.
(12) Balakrishnarajan, M. M.; Venuvanalingam, P. Unpublished results.