Computer Generation of Pauling Bond Orders Using Heuristic Search†

M. M. Balakrishnarajan and P. Venuvanalingam*

Department of Chemistry, Bharathidasan University, Tiruchirappalli 620 024, Tamilnadu, India

Received November 1, 1994⊗

A novel heuristic search algorithm is developed for the computation of Pauling bond orders (PBOs) of chemical graphs. This graph theoretic bond order is calculated from perfect matchings of graphs. As computation of perfect matching itself is a NP class problem, the complexity of the present algorithm is also exponential. Artificial intelligence techniques like heuristic scissoring and obligatory constraint seeking are used to push the tractable limit. The latter technique is especially suited for chemical graphs. This is the first computer algorithm for computation of PBOs, and the code computes the bond orders and partitions the edges according to their values. The program based on this algorithm is found to handle very large graphs within reasonable time.

1. INTRODUCTION

Computational chemical graph theory is concerned with developing computer algorithms and codes that can be used for computing certain invariants that have chemical relevance. Such invariants have been defined in chemical graph theory and have simple computational and topological origin. The increasing popularity of chemical graph theory is due to its simplicity, elegance, and extension to molecules that are considered too ambitious for quantum chemical computations. In the computation of graph theoretic invariants, the need for computerization have been emphasized in several instances.[1−14] In many occasions, pure graph theoretic methods have been proven to be restrictive; either the procedure applies to a special context[15] or if hand computations are possible they have a natural limit.[16] Further, computerization pushes the tractable limit to a wider range. It is very important to note here that such extensions to a larger range sometimes leads to conclusions that question the validity of concepts proposed based on results obtained on hand computation limits.[6,17] In other words, the concepts has to be tested in a wider range, and this can lead to more generalizations. Sometimes computerization need newer graph theoretical models as existing pure graph theoretic hand computation procedures may not always be straightforwardly adaptable for this purpose.[3] Once computerizations are envisaged, advanced techniques like artificial intelligence,[1−3] parallelization,[7,8] and vectorization[9] can be used to increase the efficiency.

In the present work, we propose a new computer algorithm to compute Pauling bond orders of chemical systems that has several chemical applications.[18−21] This is the first ever computer algorithm to compute this bond order, and the algorithm works in a more general context than the existing hand computational algorithms.[22]

The concept of bond order was introduced almost 50 years ago by Linus Pauling[23] for characterizing individual C−C bonds in polycyclic conjugated hydrocarbons. This bond order is called Pauling bond order (PBO) and is the precursor for many bond orders to be defined later.[24] PBO of a bond "e" in a molecule is defined as the ratio of the number of Kekulé structures that keeps the bond "e" as double bond to the total number of Kekulé structures of the molecule. In this equation

$$\mathrm{PBO(e)} = \frac{\mathrm{K(G\text{-}(e))}}{\mathrm{K(G)}}$$

where K(G) and K(G-(e)) are Kekulé structures of the main graph and the subgraph in which the edge "e" and its incident edges are deleted. In other words, PBO of a bond represents the relative frequency of that bond appearing as a double bond in the totality of Kekulé structures. It is also known as π bond order or mobile bond order. PBO has been compared with VB and MO bond orders. Recently Randić and Guo[24] have defined a generalized bond order (GBO) taking into account the differing contributions of various Kekulé structures of a molecule, and they have also shown that GBO reduces to PBO if all Kekulé structures are weighted equally. They have compared the GBO with other bond orders.

PBOs have many straightforward applications. Apart from characterizing a bond, it is used for obtaining RE,[21] maximal valence structures,[18,19] and bond lengths,[25,26] for expressing local aromaticity,[27] and in calculating random valence structures.[28] There are few methods to compute PBOs,[22,29−31] and this problem is also covered in a recent review.[32] It can be seen from all these reports that PBOs are computed from Kekulé structures and can also be obtained by Hückel MO using using Ham−Ruedenberg approach.[30,31] Therefore determination of PBO reduces to enumeration of Kekulé structures. For determining PBO of a bond "e", denoted as PBO(e), one would delete the bond "e" and the adjoining bonds and determine KSC of the remaining edge deleted graph. Now this is repeated for all bonds of the molecule and then PBO(e) is determined from K(G-(e)) and KSC as mentioned before. We have already shown[5] that enumeration of Kekulé structures of general graph is a NP-class problem. Now for obtaining PBOs of all bonds of G(V,E), the procedure is to be repeated for $|E|$ number of times. Moreover, there is no algorithm available in the literature, except a recent report,[2] to determine the KSC of a general graph. Most of the available reports on PBOs used either PV paths or similar methods to determine KSCs. These methods will not work for a general case. We develop here an efficient algorithm which allows one to compute PBOs

* Author to whom correspondence should be addressed.
† Dedicated to the memory of Professor Linus Pauling.
⊗ Abstract published in *Advance ACS Abstracts,* June 15, 1995.

0095-2338/95/1635-0717$09.00/0

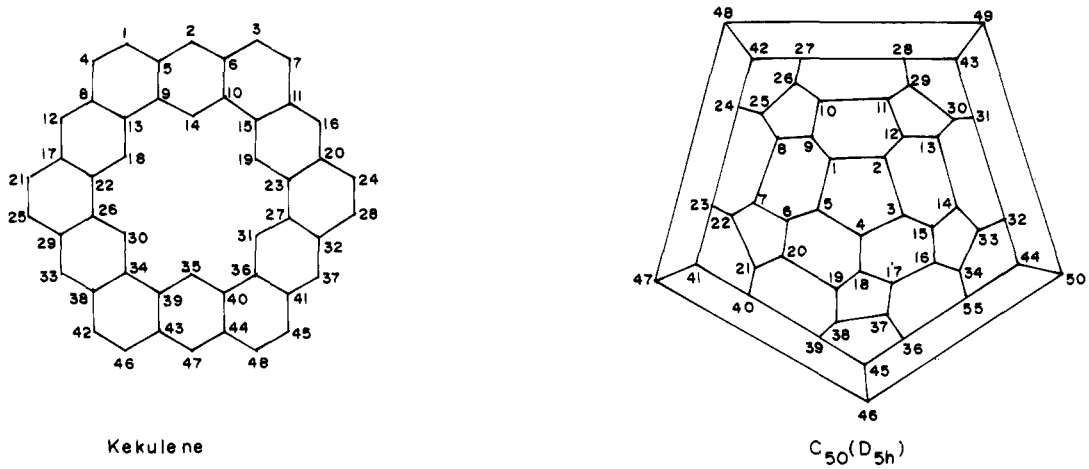

Figure 1. Labeling used for Kekulene and C_{50} (D_{5h}) (Schlegel diagram) for Pauling bond order based edge partitioning in Table 2.

of any graph straightforwardly. We illustrate the power of our method by taking two specific examples. One is a coronoid and another fullerene cluster of size 50 (Figure 1).

Our algorithm is based on search and is similar to the one that we reported earlier[2] on enumeration and generation of perfect matchings on graphs. Various heuristic techniques are used to improve the efficiency of the algorithm further.

2. ALGORITHM DESCRIPTION

In this section we introduce the basic search skeleton of our algorithm for evaluating bond orders. The search skeleton is similar to the one that is reported for generating all the perfect matchings on graphs.[2] Where earlier PBO algorithms involve determination of KSC of graphs and edge deleted graphs recursively, our algorithm enumerates K(G) as well as K(G-(e)) simultaneously. This considerably reduces the time complexity. As our algorithm applies to any graph G it has wider applicability than what existing algorithms do. The basic search skeleton applies to any general graph, and heuristic enhancements can be implemented for chemical graphs.

2.1. Efficient Graph Representation Using Edge Partitions. The abstract data structure introduced here is not commonly used to represent graphs. Here, we define a binary relation δ for the graph G of vertex set V, for unique representation of edges $\langle x,y\rangle$ such that

$$\text{For all } x,y \in V, (x \text{ adj } y) \Lambda (x < y)\} \qquad (2.1)$$

where (x adj y) is true if x and y are adjacent. This relation avoids redundancy but leaves the graph information highly fragmented. These edges are defragmented into partitions $Ç_1....Ç_n$ using another relation Ç such that

$$Ç_i = \{\langle x,y\rangle | \exists \langle x,y\rangle \in \delta, \exists i \in V, x = i\} \qquad (2.2)$$

Ç is an equivalence relation and induces n partitions in δ where n is the total number of vertices in G. Though connectivity is not apparent in this representation as it is with other standard representations like adjacency matrix, adjacency list, etc., it is found convenient for framing the search skeleton.

2.2. Problem Representation. Using the structured list of edges described above as symbols, we construct a formal system. Our earlier algorithm can be modified to compute Pauling bond orders in different ways. As reported earlier, any production system for counting the perfect matchings can be used for calculating PBO of a given bond. But, for computing the PBOs of all edges in a graph, this procedure is very cumbersome, since it is applied repeatedly for all edges one by one. Especially, for an arbitrary graph in which the Kekulé structure counting is a NP problem, demand for time will be prohibitively large even as the earlier AI algorithm is used. Alternatively, PBOs can be computed by generating and recognizing the edges individually for every Kekulé structure. This procedure will again increase the time complexity but to a lesser extent. Here we have effectively framed a modified global database so as to keep the time complexity of the system unchanged.

Global database represents the information about the various nodes of the search tree. This consists of matchings, depth, and position of pointer in the edge list, etc., corresponding to the current frame. We use the abstract data structure popularly known as Dictionary[33] to represent the matchings of various tree nodes. This is a special set theoretic structure which forbids all set operations other than Insert, Delete, and Membership. The exact edge information of the matching is lost by using Dictionary, but it is seldom required in the present approach. In addition to these structures, we need counters to enumerate the number of Kekulé structures arising from the subtrees rooted at the current frame.

Using this global database, various matchings of the search tree are formed from the edge list by using disjoint-set-union rule. Disjoint-set-union rule checks if the matching of the given node and the selected edge partition are disjoint; in which case the partition will be ignored, and search is continued in the remaining partitions. Edges of the disjoint partition are again checked for disjointness, and, if so, the vertices of the disjoint edge are inserted into the current matching to form a daughter node. The remaining edges are also tested with the current node to generate all possible successors.

Depth first control regime is used for generating the search tree. Since the depth of the search tree is a polynomial function of the graph size and its width is an exponential function, memory complexity is polynomial if depth first search is adopted. Daughter nodes are generated recursively until all the edges in the list are exhausted or if the current matching turns out to be a maximal matching. If it is a Kekulé structure, then it is counted and the control backtracks

Table 1. Algorithm in Pseudo for the Generation of All Kekulé Structures on a General Graph

```
Procedure PAULING(    POSITION : pointer to current partition
                      DEPTH    : Depth level counter
                  VAR MATCH    : matching of the current frame
                  VAR KEKTOT   : No. of Kekulé structures
                  VAR BO       : ARRAY for counting Kekulé strutures
                                 involving different edges    )
VAR
    KEKNEW : No. of Kekulé structures in the current frame

  REPEAT
    Initialize KEKTOT, KEKNEW;
    IF NOT (POSITION IN MATCH) THEN
      BEGIN
        FFDP <= POSITION
        FOR ALL <X,Y> ∈ FFDP DO
          BEGIN
            IF NOT(Y IN MATCH) DO
              BEGIN
                INSERT X and Y to MATCH
                IF MATCH is not a kekulé structure THEN
                  BEGIN
                    PAULING( SUCC(DEPTH), POSITION, MATCH, KEKNEW, BO)
                    KEKTOT <= KEKTOT + KEKNEW;
                    BO<X,Y> := BO<X,Y> + KEKNEW;
                  ENDIF
                ELSE
                  BEGIN
                    KEKTOT <= 1;
                    BO<X,Y> <= BO<X,Y> +1;
                  ENDELSE
                DELETE X and Y from MATCH
              ENDFOR
            ENDFOR
    POSITION <= POSITION + 1
  UNTIL FFDP is found or the partition list is exhausted.
END.
```

to the previous frame. Everytime the backtracking occurs, the respective number of Kekulé structures are counted, which corresponds to the edge, whose insertion gives raise to the current frame. Thus, the number of Kekulé structures arising from various edges of the graph are recorded in each frame after every backtracking. Since this involves a O(1) time the overall complexity of the inference engine remains completely unaltered regardless of the input nature.

The process described above involves ambitious generation of all matchings. But it is sufficient if we traverse through those paths of the search tree in which at least a single perfect matching is guaranteed. Thus various techniques of artificial intelligence can be adopted to identify and avoid the blind alleys in the path. Heuristic scissoring is one such technique that is described in the next section.

3. SCISSORING BLIND ALLEYS

As described in section 2.1, the graph information is obtained by treating the edges as symbols lying in different partitions $Ç_i$, $i = 1$ to $n - 1$. The search is conducted in this order of increasing i so that at any instant, the daughter nodes are generated from the edges of the higher order partitions with respect to the current position of the partition pointer. Since the number of partitions that can be disjoint to a particular matching is a function of n, the number of daughter nodes, i.e., paths descending from the current node, are also of the same complexity. But among these paths, only a few lead to Kekulé structures, and the rest of them are blind alleys. We now show how two successive improvements can be made to this solution. In this section we describe the scissoring of certain blind alleys with the guidance of heuristics so that they can be removed from the inference engine's problem space. This approach leads to the first improvement on the algorithm that is explained as follows.

When heuristic scissoring is applied, the number of edge partitions $Ç_i$ that is considered for disjoint edge search is reduced to unity, in every frame to keep the branching factor function independent of graph size. To attain this independence, heuristic selection is made among the various disjoint partitions. In each frame, the disjoint partition that lies closer to the pointer of the edge partitions is selected. Since the partitions are checked sequentially for disjointness, the first found disjoint partition, i.e., FFDP, is selected at every frame for disjoint edge search. In cases where FFDP is empty, the engine backtracks to the previous frame assuming the current node is a blind alley.

This assumption is found to be true, and these heuristics always lead to accurate results without any error. (For a systematic proof for this implication see ref 2.) The criterion used for grouping the edges in the sorted partition list (described in section 2.1) is such that the edges in all the partitions except FFDP lead to blind alleys. The selection

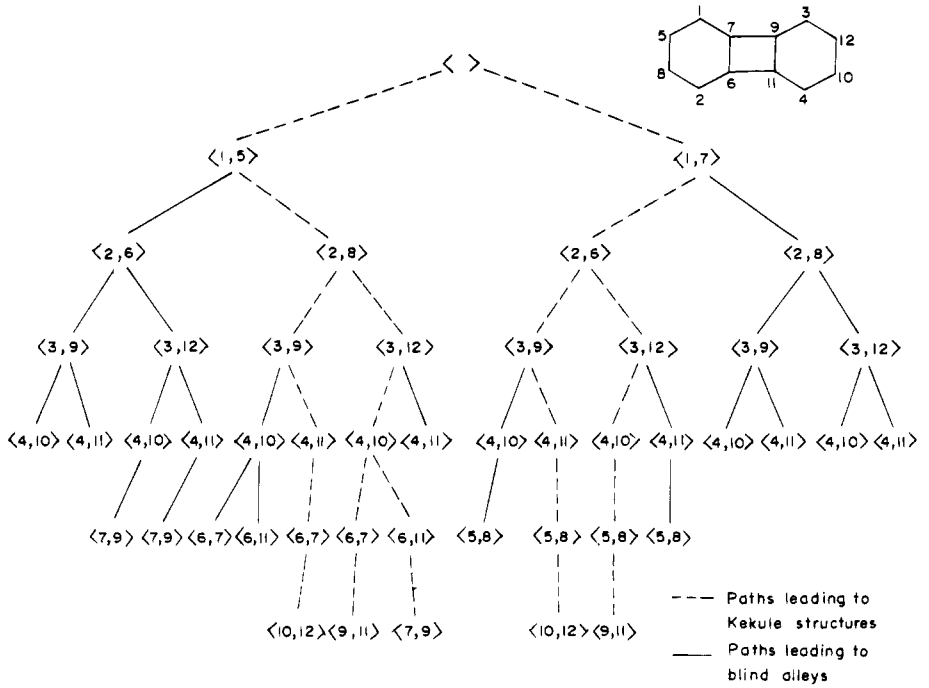

Figure 2. The search tree of the graph that is traversed during the evaluation of Pauling bond orders. Every leaf-node in the maximum depth represents a Kekulé structure.

of FFDP seldom requires additional computation time or memory since the list is sorted, and this facility can be achieved only if the input information is presented in the form as described in section 2.1. The enhanced algorithm using heuristics is given in Table 1.

The heuristic tree for a sample molecule is given in Figure 2. The impact of these heuristics is so effective that it scissors a large number of blind alleys even without taking proportionate time or additional memory. But still the NP-nature of the problem remains unaltered, and these heuristics just extend the practical size of the input that can be effectively handled. Figure 2 also shows that the blind alleys are not completely wiped out even after the implementation of these heuristics. It may be minimum, but it too can grow overwhelmingly for large graphs. Hence, we construct additional heuristics which exploit a typical nature of graphs that are of interest in chemical domain.

4. SEEKING OBLIGATORY CONSTRAINTS

Further improvements of the algorithm described above can be achieved by reducing the blind alleys that are still lying in the inference engine's problem space. This is done as follows.

Chemical graphs are very sparse and have vertices with a maximum degree of 3. The vertices with degree 1 are called pendant vertices as they are incident to pendant bonds in the graph. Pendant bonds are also called fixed bonds[34] as they are compulsory (fixed) members of all Kekulé structures. Such pendant bonds in the graphs—either in the parent graph or in the edge deleted graph at every frame of search—are included as obligatory members, herein referred to as constraints. One can note from Figures 2 and 3 that such constraints eliminates many blind alleys. For instance, in the graph referred to in Figures 2 and 3, edge ⟨1,5⟩ is selected first, and then the partition containing ⟨2,6⟩ and ⟨2,8⟩ is the FFDP. If ⟨1,5⟩ is deleted from the graph along with its incident edges, bond ⟨2,8⟩ becomes the pendant bond and hence is taken as a constraint (Figure 3). This is repeated in the subsequent frames of search. This eliminates the paths leading from edge ⟨2,6⟩, and one can understand from Figure 2 that these paths never lead to Kekulé structures. Thus the blind alleys are eliminated to the maximum extent in the search process by invoking the obligatory constraint seeking procedure.

Though pendant bonds are not abundant in chemical graphs, the process of selecting an edge in every frame and

Table 2. Pauling Bond Orders of Edges for the Labeled Graphs Given in Figure 1

chemical graph	bond order (no. of edges)	edges				
	0.8500 (6)	⟨1,4⟩	⟨3,7⟩	⟨21,25⟩	⟨24,28⟩	⟨42,46⟩
		⟨45,48⟩				
	0.5000 (24)	⟨2,5⟩	⟨2,6⟩	⟨8,12⟩	⟨9,14⟩	⟨10,14⟩
		⟨11,16⟩	⟨12,17⟩	⟨13,18⟩	⟨15,19⟩	⟨16,20⟩
		⟨18,22⟩	⟨19,23⟩	⟨26,30⟩	⟨27,31⟩	⟨29,33⟩
		⟨30,34⟩	⟨31,36⟩	⟨32,37⟩	⟨33,38⟩	⟨35,39⟩
		⟨35,40⟩	⟨37,41⟩	⟨43,47⟩	⟨44,47⟩	
Kekulene	0.3500 (12)	⟨5,9⟩	⟨6,10⟩	⟨8,13⟩	⟨11,15⟩	⟨17,22⟩
		⟨20,23⟩	⟨26,29⟩	⟨27,32⟩	⟨34,38⟩	⟨36,41⟩
		⟨39,43⟩	⟨40,44⟩			
	0.1500 (16)	⟨1,5⟩	⟨3,6⟩	⟨4,8⟩	⟨7,11⟩	⟨9.13⟩
		⟨10,15⟩	⟨17,21⟩	⟨20,24⟩	⟨22,26⟩	⟨23,27⟩
		⟨25,29⟩	⟨28,32⟩	⟨34,39⟩	⟨36,40⟩	⟨38,42⟩
		⟨41,45⟩				
	0.4417 (10)	⟨1,9⟩	⟨2,12⟩	⟨3,15⟩	⟨4,18⟩	⟨5,6⟩
		⟨41,47⟩	⟨42,49⟩	⟨43,49⟩	⟨44,50⟩	⟨45,46⟩
	0.3628 (10)	⟨7,8⟩	⟨10,11⟩	⟨13,14⟩	⟨16,17⟩	⟨19,20⟩
		⟨23,24⟩	⟨27,28⟩	⟨31,32⟩	⟨35,36⟩	⟨39,40⟩
C_{50}	0.3581 (20)	⟨7,22⟩	⟨8,25⟩	⟨10,26⟩	⟨11,29⟩	⟨13,30⟩
		⟨14,33⟩	⟨16,34⟩	⟨17,37⟩	⟨19,38⟩	⟨20,21⟩
(D_{5h})		⟨21,40⟩	⟨22,23⟩	⟨24,25⟩	⟨26,27⟩	⟨28,29⟩
		⟨30,31⟩	⟨32,33⟩	⟨34,35⟩	⟨36,37⟩	⟨38,39⟩
	0.2838 (5)	⟨21,22⟩	⟨25,26⟩	⟨29,30⟩	⟨33,34⟩	⟨37,38⟩
	0.2791 (30)	⟨1,2⟩	⟨1,5⟩	⟨2,3⟩	⟨3,4⟩	⟨4,5⟩
		⟨6,7⟩	⟨6,20⟩	⟨8,9⟩	⟨9,10⟩	⟨11,12⟩
		⟨12,13⟩	⟨14,15⟩	⟨15,16⟩	⟨17,18⟩	⟨18,19⟩
		⟨23,41⟩	⟨24,42⟩	⟨27,42⟩	⟨28,43⟩	⟨31,43⟩
		⟨32,44⟩	⟨35,44⟩	⟨36,45⟩	⟨39,45⟩	⟨40,41⟩
		⟨46,47⟩	⟨46,50⟩	⟨47,48⟩	⟨48,49⟩	⟨49,50⟩

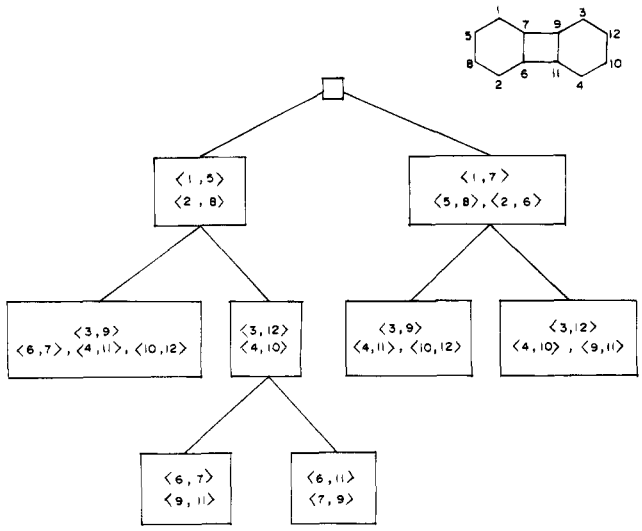

Figure 3. The search tree after seeking obligatory constraints.

subsequent deletion of the selected edge and incident edges to it, as explained earlier, can create temporary pendant bonds. Hence a search through the vertices that are incident to the current edge is made to force additional constraints. Any vertex of degree 2 incident to the current edge forms an additional constraint in the next frame. Further, this process can be repeated iteratively over the newly selected constraint edge, i.e., the search can be extended to other bivalent vertices which are incident to the current constraint edge. When all the additional constraints are found, they are included in the same frame, and the search process is continued as mentioned in the earlier section, without any modification. It is also possible that an edge choice may leave some of its incident vertices completely disconnected without any other edges incident on it. When this happens, the inference engine is forced to backtrack prematurely even with unexhausted edge partitions.

Implementation of these heuristics requires the degree of all the vertices as another data item in the global data base. After the selection of an edge the degree of vertices incident to the current edge are decreased by one at every frame. This happens because edges incident to the selected edge is also deleted in the process. Since the selection of obligatory edges are made using the degree criterion, the normal selection through the partition pointer is interrupted in every frame. But this does not add any execution time difference, because even in the absence of blind seek procedure, the same edges will be selected somewhere in the passage of the inference engine. The performance of the algorithm is significantly improved by implementing these heuristics and the time gain achieved is due to the elimination of blind alleys.

Due to the addition of obligatory edges at every frame, the choice of disjoint edges can be reduced drastically thereby removing many of the blind alleys. This can be easily verified from Figure 2. In the case of alternant systems as the one taken for illustration in Figure 2, it is found that the problem space of the inference engine is completely devoid of blind alleys[6] once these heuristics are implemented.

5. CONCLUSION

A search based symbolic manipulation algorithm has been designed to compute Pauling bond orders. This is the first computer algorithm and it applies to any chemical graph PBO of a bond that is computed as the frequency of that bond as a double bond in the totality of Kekulé structures. Kekulé structures are enumerated as disjoint sets of edge lists. As this enumeration becomes combinatorially exploding at certain limits, additional heuristics like heuristic scissoring and obligatory constraint seeking have been adopted. Thus, partially offsetting the combinatorial explosion improves the performance of the algorithm significantly. Pauling bond orders of two sample graphs using the program developed on this algorithm is also presented in Table 2.

ACKNOWLEDGMENT

We thank Professor M. Randić, Iowa for his reprints. The financial support from CSIR, India to one of us (M.M.B.) in the form of a Senior Research Fellowship is gratefully acknowledged. We thank the referees for their useful comments.

REFERENCES AND NOTES

(1) Balakrishnarajan, M. M.; Venuvanalingam, P. General Method for the Computation of Matching Polynomials of Graphs. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 1122−26.
(2) Balakrishnarajan, M. M.; Venuvanalingam, P. An Artificial Intelligence Approach for the Generation and Enumeration of Perfect Matchings on Graphs. *Computers Math. Applic.* **1995**, *29*, 115−121.
(3) Balakrishnarajan, M. M.; Venuvanalingam, P. Learning Approach for the Computation of Generalized Wheland Polynomials of Chemical Graphs. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 1113−1117.
(4) Manoharan, M.; Balakrishnarajan, M. M.; Venuvanalingam, P.; Balasubramanian, K. Topological Resonance Energy Predictions of the Stability of Fullerene Clusters. *Chem. Phys. Lett.* **1994**, *222*, 95−100.
(5) Balakrishnarajan, M. M.; Venuvanalingam, P. A Fast Graph Traversal Algorithm for the Computer Generation of P-V Paths of Benzenoid Graphs. *Comput. Chem.*, in press.
(6) Balakrishnarajan, M. M.; Venuvanalingam, P., unpublished results.
(7) Venuvanalingam, P.; Thangavel, P. Parallel Algorithm for the Computation of Characteristic Polynomials of Chemical Graphs. *J. Comput. Chem.* **1991**, *12*, 779−783.
(8) Thangavel, P.; Venuvanalingam, P. Algorithms for the Computation of Molecular Distance Matrix and Distance polynomials of Chemical Graphs on Parallel Computers. *J. Chem. Inf. Comput. Sci.* **1993**, *33*, 412−414.
(9) Balasubramanian, K. Comments on Characteristic Polynomials of a Graph. *J. Comput. Chem.* **1991**, *12*, 248−253.
(10) Balasubramanian, K. Computer Generation of Characteristic Polynomials of Graphs. *J. Comput. Chem.* **1984**, *5*, 387−394.
(11) Balasubramanian, K. and Ramaraj, R. Computer Generation of King and Color Polynomials of Graphs and Lattices and Their Applications to Statistical Mechanics. *J. Comput. Chem.* **1985**, *5*, 441.
(12) Ramaraj, R.; Balasubramanian, K. Computer Generation of Matching Polynomials of Chemical Graphs. *J. Comput. Chem.* **1985**, *6*, 122−141.
(13) Müller, W. R.; Szymanski, K.; Knop, J. V.; Trinajstić, N. An Algorithm for Construction of Molecular Distance Matrix. *J. Comput. Chem.* **1987**, *8*, 170−173.
(14) Kopecky, K. J.; Randić, M. Computer Generation of Generalized Wheland Polynomials. *Comput. Chem.* **1987**, *11*, 29−40.
(15) He, W. C.; He, W. J. Peak Vally Path Method on Benzenoid and Coronoid Hydrocarbons. *Topics Curr. Chem.* **1990**, *15*, 196−210.
(16) Randić, M.; Hosoya, H.; Ohkami, N.; Trinajstic, N. The Generalized Wheland Polynomial. *J. Math. Chem.* **1987**, *1*, 97−122.
(17) Randić, M.; El Basil, S. Graph Theoretical Analysis of Large Benzenoid Hydrocarbons. *J. Mol. Struct. (Theochem)* **1994**, *304*, 233−245.
(18) Randić, M.; Piasanski, T. On Maximal Valence Structures. *Rep. Mol. Theor.* **1990**, *1*, 107−114.
(19) Fries, K; Bicyclic Compounds and their Comparison with Naphthalene.III. *Justus Liebigs Ann. Chem.* **1927**, *454*, 121−324.
(20) Randić, M. Fully Benzenoid Systems Revisited. *J. Mol. Struct. (Theochem)* **1991**, *229*, 139−153.
(21) 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.
(22) Randić, M. Graph Theoretical Derivation of Pauling Bond Orders. *Croat. Chem. Acta* **1975**, *47*, 71−78.

(23) Pauling, L. The Nature of Chemical Bond; Cornell University Press: Ithaca, NY, 1948.
(24) Randić, M.; Guo, X. Generalized Bond Orders. *Int. J. Quantum Chem.* **1994**, *49*, 215−237.
(25) Herndon, W. C. Resonance Energies of Aromatic Hydrocarbons. Quantitative Test of Resonance Energy. *J. Am. Chem. Soc* **1973**, *94*, 2404−2406.
(26) Herndon, W. C.; Ellzey, M. L. Resonance Theory. V. Resonance Energies of Benzenoid and Non-benzenoid π-systems. *J. Am. Chem. Soc.* **1974**, *96*, 6631−6642.
(27) Randić, M. Graph theoretical approach to local and overall aromaticity of benzenoid hydrocarbons. *Tetrahedron* **1975**, *31*, 1477−1481.
(28) Randić, M. ; Solomon, V.; Grossman, S. C.; Klein, D. J.; Trinajstić, N. Resonance energies of large conjugated hydrocarbons by a statistical method, *Int. J. Quant. Chem.* **1987**, *23*, 35−59.
(29) Randić, M.; Henderson, L. L.; Stout, R.; Trinajstić, N.; Conjugation and Aromaticity in Macrocyclic Systems. *Int. J. Quantum Chem., Quantum Chem. Symp.* **1988**, *22*, 127−141.
(30) Ham, N. S.; Rudenberg, K. Mobile bond orders in conjugated systems. *J. Chem. Phys.* **1958**, *29*, 1215−29.
(31) Ham, N. S.; Rudenberg, K. Mobile bond orders in the resonance and molecular orbital theories. *J. Chem. Phys.* **1958**, *29*, 1229−31.
(32) John, P.; Sachs, H. Calculating the Number of Perfect Matchings and Spanning Trees, Pauling's Orders, the Characteristic Polynomial, and the Eigenvectors of a Benzenoid System. *Topics Curr. Chem.* **1990**, *153*, 147−180.
(33) Aho, A. V.; Hopcroft, J. E.; Ullman, J. D. Data Structures and Algorithms; Addison Wesley: New York, 1985.
(34) Chang, F.; Li, X.; Zhang, H. Hexagonal Systems with Fixed Bonds. *Disc. Appl. Math.* **1993**, *47*, 285−296.

CI9401229