Efficient Exact Solution of the Ring Perception Problem

Renzo Balducci and Robert S. Pearlman*

Laboratory for Molecular Graphics and Theoretical Modeling, College of Pharmacy, University of Texas at Austin, Austin, Texas 78712-1074

Received October 29, 1993®

The literature of the past 3 decades reports a large number of algorithms dealing with the perception of a "smallest set of smallest rings" (SSSR). However, the vast majority of the published algorithms are based upon heuristic approaches which provide an acceptable answer for some, but not all, possible connected graphs. This proliferation of approximate solutions to the SSSR perception problem arises primarily from the widespread belief that the problem is NP-complete (exponential order). Here, we present a highly efficient exact solution to this problem with a polynomial computational order. Our novel algorithm is based upon the concurrent traversal of a set of N partial breadth-first trees of the structural graph (where N is the number of nodes). The graph is represented as an object-oriented model in which each node acts as a message transceiver in a network. Each message contains the history of the connected path traversed as it propagates through the network. Message collisions correspond to ring closures. This "chain-message" algorithm detects a superset of an SSSR of the structure. The rings of this superset are identified in increasing ring size order. The SSSR is then rapidly selected from this set with a series of linear independence tests. The algorithm can be easily extended to accommodate the canonical selection of an application-defined "preferred" SSSR. We show that the order of this algorithm is *at most* $O(d^3N^2 \log N)$, where d is the maximum degree of any node of the sturcture. For chemical structures (for which $d \ll N$), this algorithm is at most $O(N^2 \log N)$. To the best of our knowledge, this is the first report of an exact solution to the ring perception problem with a computational order small enough to be useful in "real world" applications.

INTRODUCTION

The identification of cyclic substructures from connectivity information is a critical aspect in the solution of problems as diverse as the analysis of electrical circuits, analysis of communication networks, and analysis of chemical structures. Chemical applications of ring identification algorithms are also quite diverse. Software for the prediction of physical and chemical properties, suggestion of synthetic strategies, chemical database management, substructural searching, structure elucidation and three-dimensional coordinate generation all require a fast and accurate method for the identification of the "chemically meaningful" rings among the potentially large number of cyclic subgraphs embedded in the molecular structure.

Even within the past few months (e.g. ref 1), articles continue to appear in the chemical literature describing various algorithms for ring perception. However, as the following brief review of the key papers will indicate, all of these algorithms suffer one of two *serious* flaws: (i) most are heuristically based and therefore provide an incomplete, inexact solution to the problem, or (ii) some do provide an exact solution but require computation times which grow exponentially with the size of the structure. Following our brief review and on the basis of a rigorous analysis of the ring perception problem, we will present an efficient and exact solution which suffers neither of the aforementioned deficiencies.

Ring identification methods can be classified into three categories of increasing complexity:

(i) *Ring detection*: identification of a basis set of the ring space of the structure

(ii) *Ring perception*: identification of a minimal basis set of the ring space of the structure (i.e., a smallest set of smallest rings, an SSSR)

(iii) *Canonical ring perception*: identification of the "preferred" SSSR based upon a collection of application dependent ring properties

A simple ring detection method may be sufficient in some cases but the vast majority of applications requires a ring perception or a canonical ring perception method.

In 1957, Gould[2] first reported that the cycles of a graph generate a finite-dimensional linear space closed with respect to the Boolean sum (exclusive-OR) of their edges. Welch[3] recognized the need for a ring detection method that would identify a basis set of the ring space, thus allowing the exhaustive generation of all possible cycles in a structure.[4] Although other investigators have presented methods for the direct identification of all the rings without an intermediate ring detection step,[5,6] all of these exhaustive algorithms suffer from the very serious intrinsic inefficiency of generating and storing a very large number of rings. In fact, the linearity of the R-dimensional ring space implies that each linear combination (Boolean sum) of rings is a ring. Therefore, the upper bound on the total number of distinct rings in a structure is $O(2^R)$. Consequently, any method for the identification of all rings is necessarily bound to an exponential computational order. Moreover, any further processing of this exhaustive set of rings also requires computer resources (time and storage) which grow exponentially with the structure's size and complexity, rapidly reaching impractical values.

To overcome this problem, several investigators explored more compact and efficient models for representing the ring space: the "maximum proper covering set of rings",[7] the "synthetically significant rings",[8] the "chemically interesting rings",[9] and the "extended set of smallest rings",[10,11] are just a few of the empirical models proposed in the chemical

* To whom correspondence should be addressed.

® Abstract published in *Advance ACS Abstracts*, April 1, 1994.

0095-2338/94/1634-0822$04.50/0

literature to minimize the use of computer resources while providing the ring information needed in various types of chemical applications. Quite interestingly, all of these models essentially describe small supersets of the "smallest set of smallest rings" originally used in *The Ring Index*[12] and by the Wiswesser line notation.[13] The commonly accepted, although imprecise, definition of SSSR as "a basis set of rings which consists of the smallest rings that can form a basis set"[9] is equivalent to our abstract definition of "minimal basis set of the ring space"[14] and, in most cases, is also equivalent to the graph-theoretical definition of the "fundamental cycles of a minimal spanning tree"[15] (although, in general, not every SSSR corresponds to a minimal spanning tree). The SSSR concept, by any definition, minimizes the usage of computer resources while providing an accurate description of the cyclic nature of the structure and, consequently, is the most widely accepted ring model for general purpose applications. The cardinality R of an SSSR equals the dimensionality of the ring space of the structure and is given (for connected structures) by the simple equation

$$R = E - N + 1$$

where E is the number of edges (bonds) and N is the number of nodes (atoms) of the structure. The literature often refers to this quantity as the "cyclomatic" number,[16] the Frerejacque number,[9] or the "nullity" of a graph representation of the structure.[17] It was shown[18] that this relationship can be derived from Euler's polyhedron formula[19] for structures with simple polyhedral geometry. We have presented elsewhere[14,20] an unconstrained proof of the general validity of this relationship.

It was pointed out[21,22] that any given structure may have, in general, more than one SSSR. Clearly this fact introduces a potential ambiguity in the definition of which SSSR is selected to model the rings of a particular structure. Many applications may just ignore this problem and use whatever SSSR is first perceived, implicitly depending upon the ordering of the connectivity data used by the ring perception algorithm. In some cases, however, it is necessary to make an "intelligent" decision regarding which SSSR to use in further processing based upon specific (application dependent) properties of the rings involved. Unfortunately, there has been no report in the literature of any attempt to resolve this ambiguity with a canonical ring perception method. However, the problem will be addressed in this work.

The most natural approach to the ring perception problem involves (i) the identification of a set of rings which includes all the rings of one or more SSSRs and (ii) the subsequent selection of an SSSR from this initial set based upon a series of linear independence tests performed in increasing ring size order. The first step is critical since it must ensure that the initial set is a superset of at least one SSSR. (For canonical ring perception we must also ensure that the initial set includes the "preferred" SSSR as determined by application-specific criteria.) This requirement would certainly be satisfied by the exhaustive set of all simple rings, but the resulting ring perception algorithm would be bound to an unacceptable exponential computational order. This problem was recognized by several investigators[8,9] who attempted to limit the extent to which the ring space is explored with the use of simple heuristic criteria. After the identification of a (nonminimal) basis set, some (heuristically determined) linear combinations of the basis rings are tested for size and linear independence in an attempt to identify a new basis set composed of smaller rings. This procedure is recursively applied until the size of the basis rings cannot be further reduced. While in some cases this minimization converges successfully to an SSSR, clearly it is not a general solution of the ring perception problem since it does not consider all possible linear combinations of the basis rings.

Other investigators have proposed methods to reorder the structural graph according to various heuristic criteria aimed at the direct perception of an SSSR using simple graph exploration techniques ("walking" algorithms). Many of these methods[23-26] make use of search strategies empirically based upon some combination of (static) connectivity information and (dyanmic) graph traversal history. Other walking methods[21,22,27,28] rely also on a classification of ring "types" (based upon ring connectivity) to select which heuristics to apply. Even within the past year, Fan *et al.*[1] presented another walking algorithm for the direct perception of an SSSR which does not require a classification of ring types. These algorithms are all reasonably efficient, and some of them appear to yield exact solutions for most structures of chemical interest. *However*, in 1982 Deo *et al.* published a formal proof[15] that the direct perception of an SSSR using a tree walking procedure is an NP-complete problem. Therefore, this approach cannot provide an exact solution with a polynomial computational order.

This proof convinced most investigators that ring perception in general is an intrinsically NP-complete problem. However, in 1987 Horton published an algorithm[29] for the exact solution of the ring perception problem in polynomial time. Horton's method is based upon the initial identification of a shortest simple chain joining each pair of nodes. These shortest chains are then paired, according to the connectivity of their terminal nodes, yielding a set of simple rings which includes (at least) all the rings of an SSSR. The rings of an SSSR are then selected from this superset using a series of linear independence tests. The computational order of this method is $O(N^7)$ (where N is the number of nodes in the structure), and, in the author's own words, it "is a very expensive algorithm" with "considerable room for improvement". The actual execution time of Horton's algorithm could be improved in some cases with the implementation of a few widely used optimization techniques. For instance, the identification of the ring systems[9] of the structure would allow the application of the algorithm to a set of smaller substructures with independent ring subspaces. Pruning the side chains[22,23,28] or, better yet, pruning the nodes not included in any ring[24] could reduce the number of nodes to be examined. An even more drastic reduction could be achieved with the homeomorphic reduction of the structural graph,[6,16,28] at the expense of a more complex method to compute the real length of a chain. Quite obviously, however, none of these optimizations would reduce the computational order of Horton's algorithm since there exists a large class of structures which are unaffected by these techniques. Nevertheless, in spite of its impractical computational order, Horton's algorithm is a major achievement since it shows that the exact solution of the ring perception problem is indeed possible with a nonexponential computational order.

This brief survey of the ring perception literature is far from complete. More extensive reviews on the subject have been published by Gray[30] and by Downs *et al.*,[17] revealing that virtually all the previously proposed ring perception algorithms make use of some combination of the following two basic approaches: (i) *graph-theoretical* techniques, using a graph representation of the structure, for sequential exploration and manipulation of nodes and edges walking along connected paths (breadth-first[15,31] or depth-first[24] searches, graph reversals,[25,31] etc.), and (ii) *linear-algebraic* techniques,

using a matrix representation of the structure, for nonsequential exploration and manipulation of structural features (column or row exchanges to reorder the structural elements,[26,31] linear independence tests.[29] etc.).

We have presented elsewhere[14] a thorough discussion of these fundamental strategies together with an analysis of the problems associated with the design of effective ring perception algorithms based upon these classical techniques. In particular, we have shown that, quite interestingly, each of these structural representation models is well suited to solve one of the two key components of the ring perception problem: (i) the breadth-first traversal of a graph allows the direct identification of the shortest paths between any pair of nodes, and (ii) the Gaussian elimination of a matrix allows the direct evaluation of the linear independence of a list of vectors. In this work we describe a novel structural representation model designed *ad hoc* for the exact solution of the ring perception problem with an efficient hybrid of the classical techniques. Our ring perception algorithm accomplishes the following two tasks: (i) *identification* of a superset of a minimal basis set of rings of the structure through the concurrent partial traversal of N breadth-first graphs "rooted" on the N nodes of the structure and (ii) *selection* of the desired set of rings from this superset through the application of a series of linear independence tests based upon the binary Gaussian elimination method.

This novel algorithm is a generally applicable, exact solution of the ring perception problem with a worst case computational order of $\mathbf{O}(d^3N^2 \log N)$, where d is the maximum degree (connectivity) of a node. In chemical structures, $d \ll N$ and the worst case order of this algorithm is therefore only $\mathbf{O}(N^2 \log N)$ when it is used in chemical applications. To the best of our knowledge, this method is the first known ring perception algorithm which provides an *exact* solution of the problem with a computational order small enough to be useful in "real-world" applications.

Moreover, the nature of the minimal basis set of rings perceived is solely determined by the superset initially identified and by the selection criteria. Hence, with relatively simple modifications, the basic algorithm can be extended to provide the canonical perception of a "preferred" SSSR based upon a configurable collection of application dependent ring properties. With this method, the canonical SSSR is directly identified from the structural topology without the need of selecting the "preferred" rings out of a larger set. Consequently, the computational order of our canonical ring perception method is essentially identical to the order of the basic ring perception algorithm.

OVERVIEW OF THE CHAIN–MESSAGE ALGORITHM

The structural representation model that we have developed specifically to support the hybrid ring perception approach is based upon the simulation of an abstract "communication network". In this context, each node (atom) of the structure is represented by an abstract "transmitter/receiver (transceiver) device" handling a flow of messages to/from other transceiver nodes (*Tnodes*) in the network. The edges (bonds) of the structure are implicitly represented by the network's "communication channels" so that each Tnode is in direct contact only with the Tnodes representing structural nodes connected to it. The operation of all the Tnodes is synchronized by a global network controller which repeatedly switches them between a "send" state and a "receive" state until the ring perception process is complete.

The communication protocol supported by this network is based upon the (essentially) simultaneous broadcast of messages from each Tnode to all the Tnodes connected to it and ensures that each message is sent only once on any given channel (edge) of the network. More precisely, each message received by a transceiver node and not involved in any "collision" (defined below) is modified to update its traversal history and forwarded to each neighboring Tnode excluding the Tnode which most recently sent the message. Therefore, during each send phase, the messages are propagated simultaneously from each Tnode and traverse the network in breadth-first order. This peculiar method of message propagation resembles the way in which U.S. mail letters are propagated in the popular (although illegal) "chain-letter" postal game. Hence, this algorithm has been dubbed the *chain-message algorithm*.

In the receive phase, each Tnode identifies any collisions occurring among the messages received. Two messages originating from the same transmitter are said to "collide" when they reach a common receiver. (A more precise definition of message collisions and the definitions of three different collision types are presented below.) Clearly, the collision of two messages reflects the presence of two distinct communication paths between a pair of Tnodes. Hence, the union of the paths traversed by a pair of collided messages is necessarily a simple ring with size equal to the sum of the path lengths. All the simple rings identified in this way are forwarded to a global ring selector and the corresponding collided messages are not propagated any further through the network. (The slightly different handling required for the identification of odd-sized and even-sized rings is discussed in the following pages.)

At the beginning of this iterative procedure, each Tnode of the network broadcasts a single message (with length zero) to each of its neighbors. No other "blank" messages are generated during the procedure. Therefore, the simple rings identified during any given send/receive cycle are all smaller than the rings which might be identified in any subsequent cycle.

At the end of each send/receive cycle, the ring selector tests the linear independence of each newly identified ring relative to the linearly independent smaller rings perceived in the previous cycles. That is, (i) the ring selector directly applies the principle that any ring reducible to a linear combination of smaller rings is not included in any SSSR of the structure and (ii) the network communication protocol ensures that the ring selector examines every simple ring of the structure in *order of increasing ring size*. When the ring selector has identified all the $R = E - N + 1$ rings of the desired set (either a minimal basis set or the canonical minimal basis set of the ring space of the structure), it notifies the network controller that the ring perception is complete and the procedure is terminated and the multiple breadth-first traversals of the structure are truncated at this point.

RING PERCEPTION NETWORK MODEL

Path-Messages. Each *path-message* is an instance of the object type *pathMsg* which maintains a record of the path traversed while being propagated through the network (see Figure 1). The path traversed by the message is represented as a *binary edge-encoded path* (BEEP) stored in an array of E bits (the bit-vector **beep**), where E is the number of edges of the structure. According to the BEEP format, each bit corresponding to an edge of the path is set to 1_2 and every other bit is set to 0_2.

The *length* c of a path-message is the number of edges traversed by the message. It can be directly computed as the

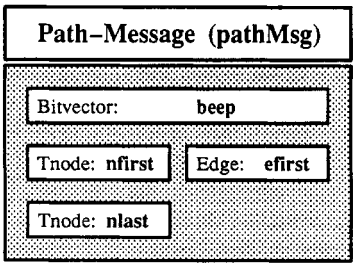

Figure 1. Data layout of the path-message object type: a bit-vector of E bits representing the *binary edge-encoded path* (BEEP) traversed by the message (where E is the number of edges of the structure), the first transceiver node (**nfirst**), the last transceiver node (**nlast**), and the first edge (**efirst**) traversed by the message.

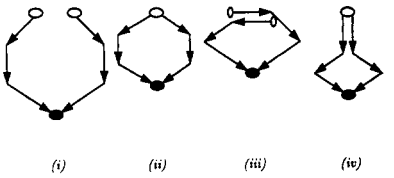

Figure 2. Path-message collisions classification: (*i*) no collision, (*ii*) node collision, (*iii*) inverse edge collision, and (*iv*) direct edge collision.

number of bits set to 1_2 in the path's **beep**. Note however that, since the length of each message is incremented by one during each retransmission and since all messages are broadcast once for each network send/receive cycle, during each cycle all messages have the same length c equal to the current value of the network iteration counter.

The pathMsg layout includes also fields for the first Tnode (**nfirst**), the first edge (**efirst**), and the last Tnode (**nlast**) traversed by the message. These last three items define the orientation of the path and, most importantly, **nfirst** and **efirst** allow the identification of collisions with other messages (*vide infra*), while **nlast** is used to prevent the retransmission of the message to its most recent sender.

Path-Message Collisions. The collision status of any two path-messages p_i and p_j with equal length c received by the same transceiver node is determined according to the following classification (see also Figure 2):

(i) no collision:

$$\mathbf{nfirst}_i \neq \mathbf{nfirst}_j \quad \text{and} \quad \mathbf{efirst}_i \neq \mathbf{efirst}_j$$

This condition arises whenever two path-messages originating from distinct transmitters reach a common receiver. Obviously, the union of the two paths does *not* include a ring.

(ii) node collision:

$$\mathbf{nfirst}_i = \mathbf{nfirst}_j \quad \text{and} \quad \mathbf{efirst}_i \neq \mathbf{efirst}_j$$

In this case, the two path-messages originated from the same transmitter and, after traversing entirely separate paths, reached the common receiver. The union of the two paths is a ring. A binary edge-encoded representation of the ring (the ring's BEER) is immediately available as the bitwise OR of the **beeps** of p_i and p_j. The size (number of edges or, equivalently, number of nodes) of the ring is $2c$. Hence, a node collision always leads to the identification of an even-sized ring.

(iii) inverse edge collision:

$$\mathbf{nfirst}_i \neq \mathbf{nfirst}_j \quad \text{and} \quad \mathbf{efirst}_i = \mathbf{efirst}_j$$

Here, p_i and p_j originated on two transmitters connected by a common edge. After traversing the common edge in opposite directions, once again the two path-messages traversed separate paths which ultimately lead to the common receiver. The union of the two paths is clearly a ring whose BEER is the bitwise OR of the **beeps** of the two paths. The size of this ring is $2c - 1$. Hence, an inverse edge collision always leads to the identification of an odd-sized ring.

(iv) direct edge collision:

$$\mathbf{nfirst}_i = \mathbf{nfirst}_j \quad \text{and} \quad \mathbf{efirst}_i = \mathbf{efirst}_j$$

In this case, p_i and p_j originated from the same transmitter and have traversed at least one common edge before taking two separate routes which converged on the common receiver. The union of these two paths is a substructure which includes at least one simple ring with size $2(c - E_x)$, where $1 \leq E_x \leq c - 2$ is the number of edges in common between the two paths. Such an even-sized ring would have been previously identified by the node collision of two path-messages with length shorter than c and the fact that it is also included in the substructure identified by a direct edge collision is of no particular consequence.

Path-Message Functions. In object oriented programming terms, the semantics of the pathMsg object type is completed by three simple maintenance and access functions.

The traversal history of each path-message is updated during each retransmission through the invocation of function push() which updates the message's **nlast** with the recipient Tnode and which sets to 1 that bit of the **beep** corresponding to the edge being traversed. Clearly, each invocation of this function increments by one the length of the path-message. This function is invoked by the Tnode's send() function.

It was mentioned above that the substructures identified by direct edge collisions may be safely ignored. However, one of the path-messages involved in each direct edge collision must be preserved for further propagation. (All other collided paths are removed from the network.) This is necessary in order to identify larger rings which might completely envelope the ring included in the substructure identified by the direct edge collision. Function merge() serves the purposes of selecting the path-message to be propagated out of a list of mutually direct-edge collided messages. The criterion for this selection may be purely arbitrary (such as: select the first path-message in the list), or it may be based upon a relative canonical ranking of the messages. This second approach is required for the canonical ring perception algorithm described further below. In either case, this function is invoked by the Tnode's receive() function.

Finally, we have seen how the rings are identified through the bitwise OR of the **beeps** of node collided or inverse-edge collided path-messages. It is advisable to encapsulate this low-level operation into a function join() which combines two such path-messages to yield the BEER of the identified ring. This function is also invoked by the Tnode receive() function.

Transceiver Nodes. As previously mentioned, each node of the structure is modeled as an abstract transmitter/receiver device implemented as an instance of the Tnode object type. A schematic layout of this object type is depicted in Figure 3. According to this design, incoming path-messages are collected in the receive buffer (which may simply be implemented as an array of path-messages). Function receive() processes these messages, detects and classifies collisions, forwards the identified rings to the ring selector, and stores the uncollided messages in the send buffer. These

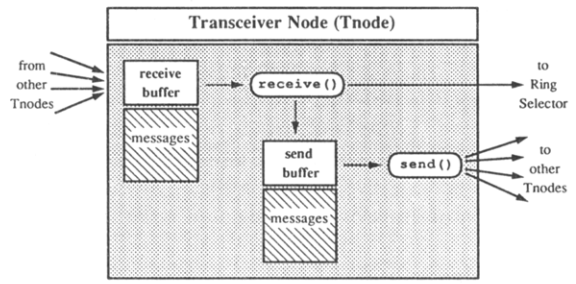

Figure 3. Schematic layout of the transceiver node (Tnode) object type.

outgoing messages are processed by function send() which broadcasts them to adjacent Tnodes. The Tnode adjacency information is directly derived from the connection table of the structure and is made accessible to the Tnodes in some form (e.g. incidence matrix, adjacency matrix, etc.).

Network Initialization. The commonly accepted definitions of the structural ring space exclude the existence of rings with size 1 or 2. Hence, there is no possibility of message collisions occurring at the first send/receive cycle of the network. Consequently, the network is initialized in a state corresponding to the beginning of the second send/receive cycle.

In this state, each Tnode n has an empty receive buffer and has d_n distinct path-messages of length 1 in the send buffer (where d_n is the degree of the Tnode). Each of these path-messages p_i is initialized as follows: (i) **nfirst** and **nlast** are both set to a Tnode n_i connected to n, (ii) **efirst** is set to the edge e_i connecting n_i to n, and (iii) all the bits of the **beep** are initialized to zero except for the bit corresponding to e_i which is set to 1.

Tnode send() Algorithm. The send() function broadcasts each path-message p stored in the send buffer of Tnode n_s to the $d_s - 1$ transceiver nodes connected to n_s which did not send p to n_s (where d_s is the degree of n_s). That is

for each Tnode n_r connected to n_s by edge e_{sr}:
  if the last Tnode (**nlast**) of p is not equal to n_r:
   update the traversal history of a copy p' of p (that is, push() n_s and e_{sr} on p');
   store this modified copy on the receive buffer of n_r

All the messages in the send buffer of n_s are processed in this way and are then deleted. Consequently, the send buffer is ultimately restored to the empty state.

Tnode receive() Algorithm. The receive() function processes the path-messages contained in the receive buffer of Tnode n_r according to their collision status as follows:

(i) merge() into a single path-message any group of path-messages exhibiting a direct-edge collision with each other
(ii) join() any pair of path-messages exhibiting an inverse-edge collision and forward the resulting BEER (an odd-sized ring) to the ring selector
(iii) join() any pair of path-messages exhibiting a node collision and forward the resulting BEER (an even-sized ring) to the ring selector
(iv) store any noncollided path-messages in the send buffer of n_r and reset the receive buffer to the empty state

An efficient implementation of this sequence of actions must be able to rapidly identify any occurrence of path-message collisions and to reliably classify the collisions according to their type. Our *optimization collision detector* makes use of two arrays of lists of path-messages as follows.

An $E \times 2$ bi-dimensional array **A** is used to identify and classify the edge collisions (where E is the number of edges of the structure). Each path-message p received by n_r is initially appended to list A_{ij} so that the **efirst** field of p is the ith edge of the structure and the **nfirst** field of p is the jth (either the first or the second) Tnode of that edge according to an arbitrary orientation. Clearly, this indexing scheme has the effect of grouping in each list of **A** all the messages with common **efirst** and **nfirst** fields thus identifing all the direct-edge collisions. Each of these lists of mutually direct-edge collided messages is then reduced to a single message by function merge(). At this point, for any index i, the path-message stored in A_{i1} (if any) and that stored in A_{i2} (if any) have a common **efirst** and different **nfirst** fields: they are a pair of messages involved in an inverse-edge collision.

The path-messages remaining after all invocations of function merge() are also tested for node collisions making use of a second array B of N lists of path-messages (where N is the number of nodes in the structure). Each path-message p is appended to list B_k so that the **nfirst** field of p is the kth Tnode of the structure. Hence, for each k the list B_k includes all the path-messages with a common **nfirst** field. Note also that the messages stored in B_k necessarily have different **efirst** fields since all groups of direct-edge collided messages have been previously merged. Therefore, every pair of messages stored in B_k is involved in a node collision.

Ring Selector Algorithm. We have seen that, during each send/receive cycle, the Tnodes forward to the ring selector the BEERs of all the rings identified during that cycle. When the ring selector is invoked at the end of the cth iteration ($c > 1$), the rings of size $\leq 2c - 2$ have already been processed (in previous iterations) and $r_{\min}$ SSSR rings have been selected among them (where $r_{\min}$ is less than the dimensionality $R = E - N + 1$ of the ring space of the structure). The following ring selection algorithm is now applied to the odd-sized rings (with size $s = 2c - 1$) identified during the cth iteration; the same algorithm is then repeated for the even-sized rings with size $s = 2c$.

(i) remove the duplicate BEERs leaving r_c unique rings
(ii) (*optional*) sort the BEERs in decreasing order of canonical "preference" (only for canonical ring perception)
(iii) for $i = 1, 2, \ldots, r_c$, if the *ith* BEER is irreducible to an exclusive-OR linear combination of the $r_{\min}$ rings previously selected,
  (a) increment $r_{\min}$ by one
  (b) append the ith ring to the basis set
  (c) if $r_{min} = R$, then STOP (that is, notify the network controller that the ring perception procedure is complete)

Each ring is redundantly identified by each of its Tnodes. Hence, for each ring of size s, s duplicate BEERs are forwarded to the ring selector. This redunancy is rapidly eliminated with a technique based upon the classic "hashing" method.[32,33] Briefly, a hash-table index is computed for each incoming BEER (based upon the bit-vector's value). If the hash-table element addressed by this index already includes the BEER, the duplicate BEER is ignored; otherwise, the BEER is included in the indexed hash-table element and in the appropriate (odd- or even-sized) BEER buffer. Through the proper choice of hashing function and hash-table size, this method approaches a perfect hashing algorithm and the removal of duplicate BEERs is achieved with an essentially linear computational order.

The actual ring selection is based upon a binary linear independence test of each new BEER against the BEERs of

previously selected linearly independent rings. The test is carried out with the use of a bit-matrix: an array of bit-vectors, maintained in upper-echelon form. (All bits below the diagonal have value zero.) This bit-matrix is initially empty. Each time a new BEER needs to be tested, a copy of the BEER is appended as an additional row of the bit-matrix. Then a Gaussian elimination procedure is carried out to convert again the bit-matrix into upper-echelon form and, most importantly, to test whether the new row bit-vector is reducible to a linear combination of the other rows. The row-vector operations performed in this test are all based upon modulo-2 arithmetic so that the vector sum is equivalent to the vector subtraction and to the bitwise exclusive-OR operation. Clearly, the Gaussian elimination of a bit-matrix with a linearly dependent row yields a bit-matrix with all the bits in the last row set to zero. In such a case, the linearly dependent BEER is ignored and the zero-ed row is removed thus leaving the upper-echelon bit-matrix ready for the next test. On the other hand, if the new BEER is linearly independent, the rank of the bit-matrix equals the number of rows ($r_{min} + 1$) and the (new) number of linearly independent rings selected so far.

The use of bit-vectors in this context is very efficient since each elementary machine operation corresponds to 32 (or 64, depending upon the hardware word size) "parallel" bit operations. The actual gain in CPU time is usually much larger than that. In fact, state of the art computer processors make use of advanced techniques (such as data and instruction caches, operand prefetch, multistage execution pipelines, vector processing units, etc.) specifically designed to greatly accelerate the execution of repeated elementary operations on contiguous words of storage. The effectiveness of these techniques is so high that, within some hardware-dependent limits on bit-vector size, the actual execution time required for an elementary operation (such as an exclusive-OR) between two bit-vectors is essentially independent of the bit-vector size. Hence, the Gaussian elimination of a bit-matrix which is already in upper-echelon form except for its last row requires an amount of CPU time which is merely proportional to the number of rows in the matrix. Since we can have at most $R = E - N + 1$ linearly independent rings in any connected structure, it follows that, on a modern computer architecture, the computational order of each linear independence test performed by the ring selector is at most $\mathbf{O}(R)$.

Network Controller: The Chain-Message Algorithm. The flow of the chain-message algorithm is globally synchronized by a single instance of the network controller object type. The responsibilities of this supervisor object include creation of a collection of N transceiver nodes representing the structure in the form of an abstract communication network, coordination of the operations of the Tnodes and of the ring selector, and collection of the list of BEERs perceived with the algorithm. The operation of the controller (and the chain-message algorithm) are summarized below:

(i) initialize the network
(ii) set the iteration counter c to one
(iii) increment c by one
(iv) invoke function send() for each Tnode
(v) invoke function receive() for each Tnodes
(vi) repeat steps iii–v until the ring selector has identified R linearly independent minimal rings

This global view allows us to briefly summarize the operations performed by the chain-message algorithm. The communication network representing the structure iterates through a series of send/receive cycles. During each cycle, a collection of messages is propagated through the network in a concurrent breadth-first fashion. The collision among the messages received by each transceiver node at each cycle provide the identification of all the simple rings of a given (progressively increasing) size. The BEERs of these simple rings are tested for linear independence and are eventually appended to the final list in increasing ring size order. Hence, the BEER of any identified simple ring is selected only if it is not reducible to a linear combination of linearly independent simple rings with smaller or equal sizes. Therefore, when the final list reaches its limiting size of $R = E - N + 1$ BEERs, it is necessarily the binary edge encoded representation of a minimal basis set of the structural ring space.

Clearly, the chain-message algorithm is an exact solution of the ring perception problem since no assumptions have been made concerning the nature of the structure or the order of the structure's node and edge lists. This approach also provides an exact solution for the canonical ring perception problem, as discussed below.

COMPUTATIONAL ORDER ANALYSIS

In the previous sections we have described a structural representation model tailored for the exact solution of the basic ring perception problem. Now, we shall estimate the maximum number of elementary operations required by this approach for the perception of a minimal basis set of rings of a structure including N nodes and $E \leq {}^1/_2 dN$ edges, where $d < \mathbf{O}(N)$ is the maximum degree of any node in the structure. Throughout this analysis we assume that the structure includes at least one ring, so that $N \leq E$.

During each send/receive cycle, the send buffer of any transceiver node only includes messages that do not collide with each other. (Any collided messages received in the previous cycle were not forwarded to the send buffer.) Hence, all the path-messages included in the send buffer of a Tnode have distinct first nodes and first edges. The number of path-messages that can possibly satisfy this constraint is at most equal to the minimum of N and E. Therefore, during each cycle the send() function of each Tnode processes at most $\mathbf{O}(N)$ path-messages. Each of these messages is broadcast to the (at most) $d - 1$ neighboring nodes which did not transmit that message. Hence, the computational order of the send() function of each transceiver node at each send/receive cycle is at most $\mathbf{O}(dN)$.

During each send/receive cycle, each Tnode sends at most $\mathbf{O}(N)$ messages to each of its neighbors. Hence, during each cycle, each Tnode may receive at most $\mathbf{O}(N)$ messages from each of its (at most) d neighbors. These (at most) $\mathbf{O}(dN)$ messages, included in the receive buffer of each Tnode, are processed by functions merge() and join() according to the collision status of each message.

These messages are first classified into direct-edge collided message groups so that the first node and the first edge shared by all the messages of each group are distinct from the first node and edge of the messages included in any other group. Therefore, the number of groups of direct-edge collided messages may be at most equal to the minimum of N and E, that is $\mathbf{O}(N)$. The messages included in each of these groups have necessarily distinct last nodes (otherwise the collision would have been detected during the previous send/receive cycle). Moreover, since the last node of any message in the recieve buffer of a Tnode represents a neighboring Tnode, each group of direct-edge collided messages may include at most $\mathbf{O}(d)$ messages. Function merge() is invoked once for each group of direct-edge collided messages and, in its basic

implementation, performs the equivalent on a single elementary operation. Hence, the computational order of function merge() for each transceiver node during each send/receive cycle is at most $O(N)$.

After merging the direct-edge collided messages, the remaining path-messages are classified in groups of at most two inverse-edge collided messages. Clearly, the first edge shared by the messages of each group is distinct from the first edge of the messages included in any other group. Hence, there can exist at most $O(E) \leq O(dN)$ groups of direct-edge collided messages in the receive buffer of each Tnode during each cycle. Function join() is invoked once for each of these groups to compute the BEER of the corresponding odd-sized ring with a single elementary bit-vector operation. Hence, the computational order of function join() for the identification of odd-sized rings during each send/receive cycle of each transceiver node is at most $O(dN)$.

The merged path-messages are also classified into node collided message groups such that the first node shared by all the messages of each group is distinct from the first node of the messages included in any other group. Therefore there may be at most $O(N)$ groups of node collided messages in the receive buffer of each Tnode during each cycle. The messages included in each of these groups have distinct last nodes (otherwise the collision would have been detected during the previous send/receive cycle). Therefore, each group of node collided messages may include at most $O(d)$ messages. Each pair of node collided messages is combined by function join() into the BEER of an even-sized ring. Each of the (at most) $O(N)$ groups of node collided messages may contribute at most $d(d-1)/2$ of these pairs. Hence, the computational order of function join() for the identification of even-sized rings during each send/receive cycle of each transceiver node is at most $O(d^2N)$.

The total amount of work done by each transceiver node during each send/receive cycle is clearly the sum of the computational orders of its four major components: send(), merge(), join() messages into odd-sized rings, and join() messages into even-sized rings. That is, according to the foregoing independent analyses, the computational order for each Tnode during each cycle is at most

$$O(dN) + O(N) + O(dN) + O(d^2N) \approx O(d^2N)$$

This analysis clearly reveals that the rate limiting component of the communcation network is the ring identification performed by function join() during the receive phase of each cycle. The upper bound of the computational order of the algorithm is thus estimated as being proportional to the maximum number of rings identifiable by each transceiver node during each cycle, $O(d^2N)$, multiplied by the number of Tnodes, N, and by the maximum possible number of cycles. This last quantity is easily estimated by considering that the largest possible simple ring would have size N. Hence, the number of cycles necessary to identify the largest ring is (at most) $O(N/2)$. Therefore, the total computational order of the communication network is *at most* $O(d^2N^3)$.

This computational order analysis is certainly correct in all respects. However, the upper bound for the order determined in this fashion appears to be unrealistically conservative. In fact, at least intuitively, the existence of any structure including $O(d^2N^2)$ rings for each size between 3 and N is somewhat questionable, even allowing for the multiple copies of each ring that are actually identified by the network. Since such a hypothetical worst-case structure appears to be "unreachable", we can begin to acquire a feel for a more realistic estimate of the computational order of the network by examining two extreme (but "reachable") worst-case examples in which each node is included in at least one ring.

(i) The minimum degree of any node in a structure in which each node is included in at least one ring is clearly 2. Any structure in which all nodes have degree $d = 2$ necessarily includes a single ring with size N. Perception of this single ring would require $N/2$ send/receive cycles and the only cycle detecting any message collisions would be the last one. These collisions would lead to the identification of N copies of the only ring with size N. Hence, in this example the network would perform a number of elementary operations proportional to $O(N)$.

(ii) At the other extreme, the maximum possible degree of any node in a structure is $d = N - 1$. Any structure in which all nodes have this maximum degree is characterized by the fact that each of its nodes is connected to all the other nodes. Clearly, any SSSR of such a structure includes only rings of size 3 and the corresponding communication network model has to perform exactly one send/receive cycle in order to identify all possible 3-sized rings. The number of 3-sized rings included in such a structure can be easily computed as the number of combinations of N nodes taken three at a time; that is,

$$\binom{N}{3} = \frac{N(N-1)(N-2)}{6} = \frac{Nd(d-1)}{6}$$

Since the network identifies three copies of each ring, the number of elementary operations performed in this maximally connected example is proportional to $O(d^2N)$.

In both of these extreme cases the actual computational cost of the ring identification procedure is significantly lower than the previously estimated upper bound. In the first case, a small number of large rings is identified; in the second case, a large number of small rings is identified. This observation supports our intuitive conjecture that there may exist an inverse relationship between the total number of rings identified and the distribution of ring sizes, although we are currently unable to precisely quantify such a relationship. Nevertheless, we are able to derive a more realistic estimate of the upper bound for the order of the chain-message algorithm with the following analysis based upon the maximum number of simple rings that may exist in any "real" structure rather than our original analysis based upon the unreachable maximum number of rings identifiable by each Tnode at each send/receive cycle.

For any structure, the maximum possible number of simple chains (paths) of any length with the same terminal pair (TP) of nodes and with no other nodes in common is clearly $O(d)$. [The terminal pair of nodes of a simple (nonbranched) chain is defined as the pair composed of the first node and the last node of the path traversing the chain in either direction.] If all these simple chains had the same length, the corresponding path-messages would all be received during the same send/receive cycle of the communication network and would all exhibit node collisions with each other, thereby leading to the identification of $O(d^2)$ even-sized simple rings. On the other hand, if these simple chains did not have equal lengths, the number of node collided pairs of the corresponding path-messages, received in separate send/receive cycles, would clearly be less than $O(d^2)$. The path-messages corresponding to any other simple chain with the same TP and with some other node(s) in common would either exhibit a direct edge collision at the receiver node (which is one of the two nodes of the TP) or would not reach the receiver because of a collision detected during an earlier send/receive cycle. The total

number of pairs of nodes is obviously $O(N^2)$; hence, the maximum number of even-sized simple rings that may be identified in any structure is $O(d^2N^2)$ where, clearly, this estimate also includes all the s duplicates identified for each ring of size s.

Similarly, for any edge e and any node n, the maximum number of simple chains of any length including e at one end and n at the other end and with no other node in common is $O(2d)$. If all these simple chains had the same length, the corresponding path-messages would be received by the Tnode representing n during the same send/receive cycle. However, in this case, the $O(2d)$ path-messages would be classified into two groups of at most $O(d)$ direct-edge collided messages, each group would be merged into a single path-message, and the two remaining inverse-edge collided messages would be joined into a single BEER with odd ring size. On the other hand, if each path-message of a given length exhibited an inverse-edge collision with another message (necessarily of the same length) and if no other path-message with the same length existed between e and n, the number of odd-sized rings identified for that particular pair of one edge and one node would be at most $O(d)$. This last scenario may exist for each of the $O(NE)$ pairs of one edge and one node of the structure. Hence, the maximum number of odd-sized simple rings that may be identified in any structure is

$$O(dNE) \approx O(d^2N^2)$$

where again this estimate also accounts for the identification of multiple copies of each ring.

It was shown above that the computational order of the communication network operations is proportional to the number of simple rings identified; hence, a realistic estimate of the upper bound of this computational order is $O(d^2N^2)$. This is also an estimate of the maximum possible number of BEERs forwarded to the ring selector. We have mentioned above that the removal of duplicate BEERs based upon a perfect hashing algorithm has an approximately constant computational order per BEER examined. Hence, the contribution of the duplicate removal to the computational order of the chain-message algorithm is at most $O(d^2N^2)$.

It was also shown that, in any structure, the maximum possible number of rings of size three (including all the duplicates) is $O(d^2N)$. Any ring of size greater than three can always be divided into two or more rings of size three. Hence, the maximum number of rings of any size cannot exceed the maximum number of rings of size 3. Therefore, the maximum number of simple rings of any given size identified by the communication network is less then or equal to $O(d^2N)$, where, for each ring of size s, there are s duplicate BEERs which are then removed. Note that after removal of the s duplicates of each ring of size s, only $1/s$ times as many rings of size s remain to be tested for linear independence. Hence, the maximum total number of distinct (nonduplicate) BEERs is the sum of the maximum number of rings of each size divided by the ring size. That is:

$$O(d^2N)\sum_{s=3}^{N}\frac{1}{s} < O(d^2N)\int_3^N \frac{1}{s}\,ds < O(d^2N \log N)$$

It was shown that, on current state of the art computer architectures, the linear independence test requires at most $O(R) \approx O(dN)$ elementary operations for each BEER tested. Therefore, the total computational cost of these tests is at most $O(d^3N^2 \log N)$.

Summarizing, a more realistic upper bound of the computational order of the chain-message algorithm is obtained from the sum of the more realistic maximum computational orders of the communication network and of the ring selector:

$$O(d^2N^2) + O(d^2N^2) + O(d^3N^2 \log N) \approx O(d^3N^2 \log N)$$

where $d < N$ is the maximum node degree. However, in chemical structures the maximum degree d is always much smaller than the number of nodes. (In most organic chemical structures, $d = 4$.) Therefore, in chemical applications the maximum computational order of this algorithm is just $O(N^2 \log N)$.

CANONICAL RING PERCEPTION

We have previously stated that the objective of a canonical ring perception algorithm is the identification of a "preferred" minimal basis set of the ring space of the structure based upon a collection of application dependent "preference" criteria. For example, in a chemical application it may be desirable to perceive an SSSR which, in addition to the minimal ring size constraint intrinsic in any SSSR, also satisfies some chemically significant criteria, such as

(i) each ring of the SSSR includes atoms (nodes) with the largest (or with the smallest) possible degree
(ii) each ring of the SSSR includes atoms with the largest (or with the smallest) possible atomic number
(iii) each ring of the SSSR includes the largest (or the smallest) possible fraction of heteroatoms
(iv) each ring of the SSSR includes the largest (or the smallest) possible fraction of chiral atoms (or of atoms with a specified chirality)
(v) each ring of SSSR includes the largest (or the smallest) possible fraction of double (or aromatic) bonds (edges)

Clearly, this is just a small sample of the types of preference criteria which may be of interest in a chemical application. Optionally, these localized node and edge properties may be used to compute canonical ranks which account also for differences in the topological environment of each node and edge. Morgan's classic "extended connectivity" method[34] and the SMILES[35] canonicalization method derived by Weininger[36] can be easily adapted to obtain a canonical environmental ranking of nodes and edges. These unique ranks can then be used as preference criteria in the canonical ring perception. Analogous collections of criteria may be defined for any type of nonchemical application in order to unambiguously specify the ring properties associated with the preferred SSSR with a virtually unlimited degree of flexibility.

In any case, the desirable or undesirable features of the SSSR rings are always specified in terms of intrinsic node or edge properties and do not depend on any particular representation (graph) of the structure. In other words, the preference criteria are always based upon a collection of *graph invariants* of the structure. The graph invariants of each node of the structure are encoded into a single composite invariant indicating the preference of the application for each particular node. These composite invariants are then used to compute a relative ranking of the nodes. The node ranks are then combined with the edge invariants to obtain the composite invariants associated with each edge. The edges of the structure are then sorted in nondecreasing order of their composite invariant values.

The relative preference among substructures with equal number of edges now can be determined based upon the relative

preference for the edges of each substructure. The binary edge, encoding of substructures into bit-vector representations greatly simplifies this task. In fact, as a consequence of the invariant based sorting of the edges, the highest order bit of such a bit-vector represents the presence or absence (value 1_2 or 0_2, respectively) in the substructure of the edge most highly preferred. Similarly, the second highest order bit corresponds to the second most highly preferred edge and so on, down to the lowest order bit corresponding to the least preferred edge. Hence, the relative preference among two substructures is directly reflected by the relative magnitude of the words (integer values) comprising their binary edge-encoded representations compared in decreasing order of numerical significance.

Our algorithm for the canonical perception of the preferred SSSR is derived from the basic chain-message algorithm with just two simple modifications which make use of this method for the determination of the relative preference of substructures based upon a collection of application dependent graph invariants. In fact, all the substructures (simple chains and rings) handled by the chain-message algorithm are represented in binary edge-encoded form (BEEPs and BEERs) relative to the edge list of the structure: the BEEPs are encoded in this way with the path-message's function push() and the BEERs are always computed as the bitwise OR of two BEEPs. Hence, if the edge list of the structure is sorted in increasing order of preference as described above, we can directly establish the relative preference for any group of BEEPs (or BEERs) with equal number of edges by comparing the numerical values of the bit-vectors. An analysis of the chain-message algorithm reveals that the nature of the perceived SSSR is solely determined by the choices made in two critical portions of the algorithm: the selection of one path-message out of each group of direct edge collided messages (performed with function merge()) and the order in which the ring selector processes each list of equal sized BEERs through the linear independence tests. In both cases, the bit-vectors involved represent substructures with equal number of edges and their relative preference can therefore be directly determined:

(i) Function merge() now must select the preferred path-message among the group of direct-edge collided messages it processes at each invocation. This is easily accomplished by selecting the path-message whose BEEP has the largest numerical value.

(ii) The ring selector now must test for linear independence the BEERs of each list in decreasing order of relative preference. In this way, the most preferred linearly independent BEERs are selected for inclusion in the canonical SSSR before testing the linear independence of the less preferred BEERs. This is accomplished by sorting each list of equal sized BEERs in decreasing order of their numerical values before proceeding with the sequence of linear independence tests, which are performed as previously described. (Recall that the SSSR is perceived in increasing ring size order.)

The upper bound for the computational order of this canonical ring perception algorithm may be estimated as follows. The initial ranking of the nodes of the structure based upon the composite node invariants can be rapidly performed using the standard "quicksort" algorithm[33] with an average computational order: $\mathbf{O}(N \log N)$. Similarly, the initial sorting of the edge list of the structure can be performed using quicksort with an average number of elementary operations proportional to

$$\mathbf{O}(E \log E) \approx \mathbf{O}(dN \log N)$$

The modified function merge() must examine the numerical values of the BEEPs of the (at most) $\mathbf{O}(d)$ path-messages included in each direct-edge collided group for each node during each send/receive cycle. However, the maximum computational cost associated with these operations never exceeds the far greater cost of examining every possible pair of inverse-edge collided and direct-edge collided messages performed with function join() in order to identify the simple rings of the structure. Therefore, the maximum total computational order of the communication network is unaffected by the modification described for function merge() and remains (at most) proportional to the number of rings identified: $\mathbf{O}(d^2N^2)$.

The modified ring selector sorts each list of (at most) $\mathbf{O}(d^2N/s)$ BEERs with size s before proceeding with the usual sequence of linear independence tests. Assuming once again that each sorting is performed with the efficient quicksort algorithm, the associated total computational cost is at most:

$$\sum_{s=3}^{N} \frac{\mathbf{O}(d^2N)}{s} \log\left(\frac{\mathbf{O}(d^2N)}{s}\right) \approx \mathbf{O}(d^2N) \sum_{s=3}^{N} \frac{1}{s} \log\left(\frac{N}{s}\right)$$

$$= \mathbf{O}(d^2N) \sum_{s=3}^{N} \frac{1}{s} (\log N - \log s)$$

$$< \mathbf{O}(d^2N \log N) \sum_{s=3}^{N} \frac{1}{s}$$

$$< \mathbf{O}(d^2N(\log N)^2)$$

The total maximum computational order of the canonical chain-message algorithm is the sum of the maximum orders of its components, that is

$$\mathbf{O}(N \log N) + \mathbf{O}(dN \log N) + \mathbf{O}(d^2N^2) + \mathbf{O}(d^2N(\log N)^2) + \mathbf{O}(d^3N^2 \log N) \approx \mathbf{O}(d^3N^2 \log N)$$

Hence, according to this analysis, the modifications of the basic chain-message algorithm required to achieve the canonical ring perception do not add any significant overhead to the estimated maximum computational order.

CONCLUSIONS

The collection of algorithms presented in this work clearly provides an exact general solution to the long standing ring perception problem. In particular, this solution does not rely upon constraints on the nature of the structure nor on specific orderings of the nodes and edges of the structural representation. Furthermore, our method does not make use of any heuristic approximations which are so common in previously proposed partial solutions of the problem. The rigorous correctness of the chain-message algorithm, its great efficiency (maximum computational order of $\mathbf{O}(d^3N^2 \log N)$, computational order in chemical structures limited to $\mathbf{O}(N^2 \log N)$), the facile adaptability for canonical ring perception, and its ease of implementation in any object oriented programming language appear to make it particularly useful for applications in chemistry as well as in many other disciplines.

ACKNOWLEDGMENT

This work was supported in part with grants from the U.S. Environmental Protection Agency, the R. A. Welch Foundation, and Tripos Associates, Inc.

REFERENCES AND NOTES

(1) Fan, B. T.; Panaye, A.; Doucet, J. P.; Barbu, A. Ring Perception. A New Algorithm for Directly Finding the Smallest Set of Smallest Rings for a Connection Table. *J. Chem. Inf. Comput. Sci.* **1993**, *33*, 657–662.
(2) Gould, R. The Application of Graph Theory to the Synthesis of Contact Networks. *Proceedings of an International Symposium on the Theory of Switching (April 2–5, 1957)*; The Annals of the Computation Laboratory of Harvard University, Annals No. 29; Harvard University Press: Cambridge, MA, 1959; pp 244–292.
(3) Welch, J. T. A Mechanical Analysis of the Cyclic Structure of Undirected Linear Graphs. *J. Assoc. Comput. Mach.* **1966**, *13*, 205–210.
(4) Gibbs, N. E. A Cycle Generation Algorithm for Finite Undirected Linear Graphs. *J. Assoc. Comput. Mach.* **1969**, *16*, 564–568.
(5) Tiernan, J. C. An Efficient Search Algorithm to Find the Elementary Circuits of a Graph. *Commun. Assoc. Comput. Mach.* **1970**, *13*, 722–726.
(6) Balaban, A. T.; Filip, P.; Balaban, T. S. Computer Program for Finding All Possible Cycles in Graphs. *J. Comput. Chem.* **1985**, *6*, 316–329.
(7) Corey, E. J.; Wipke, W. T. Computer-Assisted Design of Complex Organic Syntheses. *Science* **1969**, *166*, 178–192.
(8) Corey, E. J.; Petersson, G. A. An Algorithm for Machine Perception of Synthetically Significant Rings in Complex Cyclic Organic Structures. *J. Am. Chem. Soc.* **1972**, *94*, 460–465.
(9) Wipke, W. T.; Dyott, T. M. Use of Ring Assemblies in a Ring Perception Algorithm. *J. Chem. Inf. Comput. Sci.* **1974**, *15*, 140–147.
(10) Downs, G. M.; Gillet, V. J.; Holliday, J. D.; Lynch, M. F. Theoretical Aspects of Ring Perception and Development of the Extended Set of Smallest Rings Concept. *J. Chem. Inf. Comput. Sci.* **1989**, *29*, 187–206.
(11) Downs, G. M.; Gillet, V. J.; Holliday, J. D.; Lynch, M. F. Computer Storage and Retrieval of Generic Chemical Structures in Patents. 9. An Algorithm To Find the Extended Set of Smallest Rings in Structurally Explicit Generics. *J. Chem. Inf. Comput. Sci.* **1989**, *29*, 207–214.
(12) Patterson, A. M.; Capell, L. T.; Walker, D. F. *The Ring Index*, 2nd ed.; American Chemical Society: Washington, D.C., 1960.
(13) Smith, E. G. *The Wiswesser Line-Formula Chemical Notation*; McGraw-Hill: New York, 1968.
(14) Balducci, R. Symbolic Structural Modeling. Ph.D. Thesis, University of Texas at Austin, Aug 1992.
(15) Deo, N.; Prabhu, G. M.; Krishnamoorthy, M. S. Algorithms for Generating Fundamental Cycles in a Graph. *ACM Trans. Math. Software* **1982**, *8*, 26–42.
(16) Matyska, L. Fast Algorithm for Ring Perception. *J. Comput. Chem.* **1988**, *9*, 455–459.
(17) Downs, G. M.; Gillet, V. J.; Holliday, J. D.; Lynch, M. F. Review of Ring Perception Algorithms for Chemical Graphs. *J. Chem. Inf. Comput. Sci.* **1989**, *29*, 172–187.
(18) Elk, S. B. Derivation of the Principle of Smallest Set of Smallest Rings from Euler's Polyhedron Equation and a Simplified Technique for Finding This Set. *J. Chem. Inf. Comput. Sci.* **1984**, *24*, 203–206.
(19) Courant, R.; Robbins, H. *What is Mathematics*? Oxford Unviersity Press: London, U.K., 1941; Chapter V, pp 235–271.
(20) Balducci, R.; Pearlman, R. S., Symbolic Structural Algebra. *J. Chem. Inf. Comput. Sci.* Manuscript in preparation.
(21) Zamora, A. An Algorithm for Finding the Smallest Set of Smallest Rings. *J. Chem. Inf. Comput. Sci.* **1976**, *16*, 40–43.
(22) Roos-Kozel, B. L.; Jorgensen, W. L. Computer-Assisted Mechanistic Evaluation of Organic Reactions. 2. Perception of Rings, Aromaticity, and Tautomers. *J. Chem. Inf. Comput. Sci.* **1981**, *21*, 101–111.
(23) Esack, A. A Procedure for Rapid Recognition of the Rings of a Molecule. *J. Chem. Soc., Perkin Trans. 1* **1975**, 1120–1124.
(24) Schmidt, B.; Fleischhauer, J. A Fortran IV Program for Finding the Smallest Set of Smallest Rings of a Graph. *J. Chem. Inf. Comput. Sci.* **1978**, *18*, 204–206.
(25) Gasteiger, J.; Jochum, C. An Algorithm for the Perception of Synthetically Important Rings. *J. Chem. Inf. Comput. Sci.* **1979**, *19*, 43–48.
(26) Hendrickson, J. B.; Grier, D. L.; Toczko, A. G. Condensed Structure Identification and Ring Perception. *J. Chem. Inf. Comput. Sci.* **1984**, *24*, 195–203.
(27) Qian, C.; Fisanick, W.; Hartzler, D. E.; Chapman, S. W. Enhanced Algorithm for Finding the Smallest Set of Smallest Rings. *J. Chem. Inf. Comput. Sci.* **1990**, *30*, 105–110.
(28) Baumer, L.; Sala, G.; Sello, G. Ring Perception in Organic Structures: A New Algorithm for Finding SSSR. *Comput. Chem.* **1991**, *15*, 293–299.
(29) Horton, J. D. A Polynomial-Time Algorithm To Find the Shortest Cycle Basis of a Graph. *SIAM J. Comput.* **1987**, *16*, 358–366.
(30) Gray, N. A. B. *Computer-Assisted Structure Elucidation*; John Wiley & Sons: New York, 1986; Chapter IX.E, pp 312–324.
(31) Ito, T.; Kizawa, M. The Matrix Rearrangement Procedure for Graph-Theoretical Algorithms and Its Application to the Generation of Fundamental Cycles. *ACM Trans. Math. Software* **1977**, *3*, 227–231.
(32) Knuth, D. E. Sorting and Searching. *The Art of Computer Programming*; Addison-Wesley: Reading, MA, 1973; Vol. 3.
(33) Sedgewick, R. *Algorithms in C*. Addison-Wesley: Reading, MA, 1990.
(34) Morgan, H. L. The Generation of a Unique Machine Description for Chemical Structures-A Technique Developed at Chemical Abstracts Service. *J. Chem. Doc.* **1965**, *5*, 107–113.
(35) Weininger, D. SMILES, a Chemical Language and Information System. 1. Introduction to Methodology and Encoding Rules. *J. Chem. Inf. Comput. Sci.* **1988**, *28*, 31–36.
(36) Weininger, D.; Weininger, A.; Weininger, J. L. SMILES. 2. Algorithm for Generation of Unique SMILES Notation. *J. Chem. Inf. Comput. Sci.* **1989**, *29*, 97–101.