A redundancy eliminating approach to linearly independent rings selection in the ring perception problem

G. Mancini
Infm, UdR Camerino.
Dipartimento di Matematica e Fisica Università di Camerino.
Via Madonna delle Carceri, 62032 Camerino (MC).
Italia.

Abstract

Based on a recently published efficient, exact algorithm to solve the ring perception problem, a new approach is presented to feed the linear independence test on rings to enter a minimal basis no duplicate information, thus reducing calls to the most demanding procedure in terms of computational order. The efficiency of a perfect hashing algorithm is actually met by a "pre-filtering" method derivated from simple considerations.

Ring perception, SSSR, Condensed Matter: Structure, Chemical Physics.

November 30, 1999

1 Introduction

Due to the availability of greater and greater computing power, molecular dynamics simulations are increasingly used as a valuable cross-confirmation tool in various fields. Noticeably, molecular dynamics simulations allow the determination of the states in which a simulate system is at any stage of its evolution process, providing information for comparison to experimental results or confirmation of theoretical models. A common starting point for further processing and analysis based on data from molecular dynamics simulations is represented by the reconstruction of the molecular structure of the simulated systems once atoms positions are given. To detect the presence of molecules and determine their structures, it is convenient to represent the system as an undirected, simple graph, associating each atom to a node and each bond between two atoms to an edge joining the respective nodes, so that a molecule in the system corresponds to a ring (cycle, loop) in the graph. A great deal of approaches to the determination of rings in a graph have been proposed during the past years ranging from heuristic, incomplete, inexact, fast algorithms to exact, slow ones whose computation time increases exponentially with the size of the graph, and the exact, efficient algorithm by Balducci and Pearlman [1] for the determination of a minimum basis (a smallest set of smallest rings, or SSSR) of the ring space of the structure. In their paper, Balducci and Pearlman used the communication network analogy letting "path-messages", conveying information on their walk through the network, propagate by means of transceivers and communication channels during alternate, synchronous "send" and "receive" states of the network; the rings perception (i.e., SSSR identification) is carried out by classifying path-messages collisions on a common receiver. Following Balducci and Pearlman's guidelines to implement a ring perception algorithm, the author has developed a new approach (called ***pre-filtering***) to handle and eliminate redundant information processing while selecting rings to enter an SSSR.

2 Basic definitions

- An *edge sequence* of length (size) $l-1$ in a graph is a finite sequence of edges (i_1,i_2), (i_2,i_3), (i_3,i_4), ..., (i_{l-1},i_l), where each edge is denoted by the pair of nodes it connects. If i_1=i_l or $i_1 \neq i_l$ the edge sequence is *closed* or *open*, respectively.
- An edge sequence of distinct edges constitutes an *edge train*.
- An open edge train in which all nodes are distinct is called a *path*.
- A closed edge train in which all nodes are distinct is called a *ring*.
- The symmetric difference or ring sum of two rings A, B is defined as $A \oplus B = (A \cup B) \setminus (A \cap B)$.

Graphs considered in this paper are simple, finite and undirected; although not strictly necessary, it furthermore is supposed they are connected, so that they contain $R = E - N + 1$ linearly independent rings, where E and N are the numbers of edges and nodes, respectively.

3 Process overview

Given a graph containing N nodes and E edges representing the system, two main tasks are performed:

1. identification of a superset of a minimal basis of rings;
2. selection of a set of R=E-N+1 rings to form an SSSR.

The graph is considered as a synchronous communication network associating each node to a transceiver, each edge to a communication channel and each path to a "path-message". All transceivers, during alternating send/receive cycles, simultaneously propagate path-messages through the communications channels linking adjacent nodes in such a way that any message originating from a node m and received by a node n is forwarded to all nodes connected to n but m. The entire process is such that, at any stage, all path-messages are the same length, this length being increased each time messages are retransmitted, so that rings are identified and processed in increasing length order to build up an SSSR. Basically, the whole process consists of four phases:

1. initialise the communication network;
2. send path-messages;
3. receive path-messages;
4. select rings;

and develops cycling over 2 through 4 until an SSSR is obtained. A simplified description of the send and receive phases is given in fig.1*(i)*,*(ii)*.

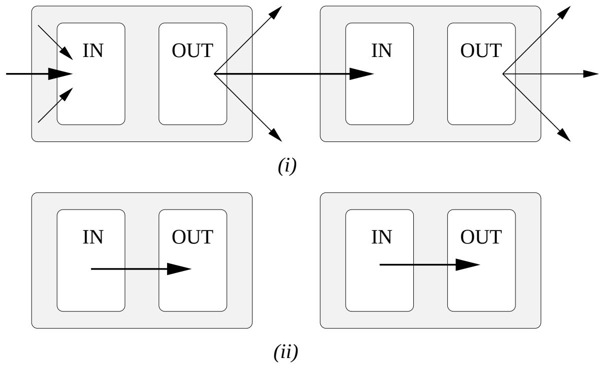

Figure 1: Sending *(i)* and receiving *(ii) path-messages*.

4 Pre-filtering

Once a ring has been identified, it must undergo a linear independence test against the elements of the

SSSR determined so far in order to enter it or be discarded. As it is clear from the detailed description of the algorithm given in [1], a ring of length L may, in principle, be built as much as L times (once per component node), thus, the same L-sized ring may be submitted up to L times to the linear independence test. Balducci and Pearlman proposed to use the well known “hashing” method [1, 2, 3] to avoid multiple linear independence tests on the same ring. On the assumption of a suitable choice of the hashing function together with a well dimensioned associated hash table, the method approaches a perfect hashing algorithm. A new method (*pre-filtering*) is now presented to substitute the hashing one, ensuring that each ring is checked for linear independence exactly once.

Let us briefly anticipate how rings are built up from path-messages collisions. Think of path-messages as messages describing the time evolution of paths through the communication network and define a collision of paths messages as their arrival on the same transceiver at the same time. As it has been mentioned above the communication network operations are driven so that at any stage all path-messages are the same length, be it l. It is easily seen (fig.2) that the union of two entirely separate (both edge-

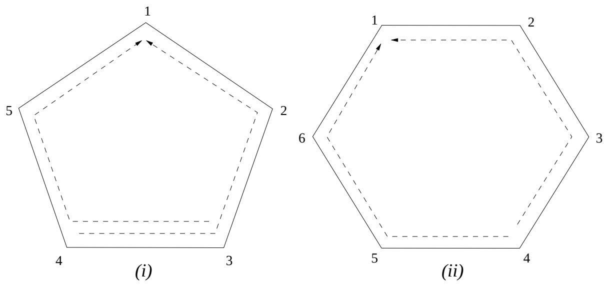

Figure 2: odd *(i)* and even *(ii)* rings construction.

disjoint and node-disjoint) path-messages- save their first edge, traversed in opposite directions - colliding on a node n gives a ring of length $2l$-1, while the union of two entirely separate path-messages - save their first node - closing on a node n gives a ring of length $2l$. Paths coupling to form rings will be indicated as *rings identification process*.

Upward selection. Consider fig.3(i): two distinct,

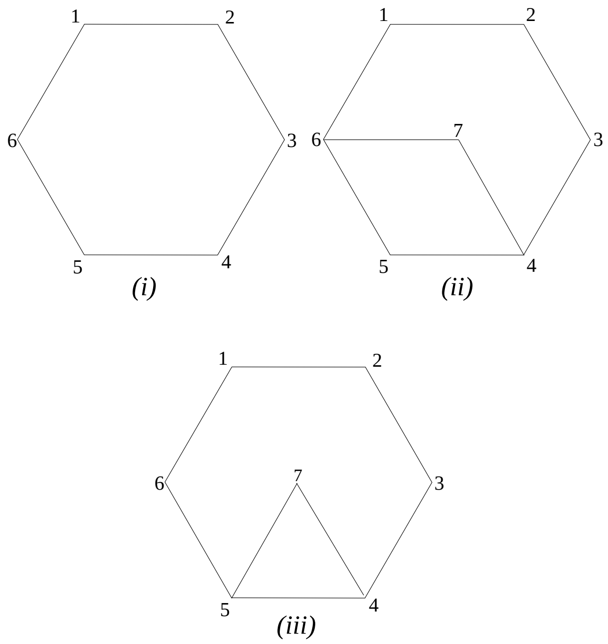

Figure 3: unforced *(i)*,*(iii)* and forced *(ii)* full formation.

3-lengthed paths (4,5), (5,6), (6,1) and (4,3), (3,2), (2,1) originating from node 4 close on node 1 to form the 6-lengthed ring (1,2), (2,3), (3,4), (4,5), (5,6), (6,1); similarly paths (5,6), (6,1), (1,2) and (5,4), (4,3), (3,2) close on node 2 to form the ring (2,3), (3,4), (4,5), (5,6), (6,1), (1,2) and so on for the remaining nodes. The formation of a ring on all of its nodes will be indicated as *full formation*. The appearence of *full formation* in the communication network depends on how paths messages are propagated: according to [1], once the paths (4,5), (5,6) - (4,7), (7,6) - (6,5), (5,4) - (6,7), (7,4) (fig.3(ii)) are used to build up the rings (4,5), (5,6), (6,7), (7,4) and (6,7), (7,4), (4,5), (5,6) , the corresponding paths messages undergo no further propagation. In this case *full formation* doesn't take place (no ring closes on nodes 1 and 3).

Full formation suggests how to devise a strategy to send the independence test no more than one L-sized ring out of the L possible equal ones. Using an arbitrary numbering for nodes, suppose *full formation* is enforced and all the rings closing on node 1 have been processed. When operating on node 2, all rings closing on it after passing through node 1 can be safely ignored since they have been already identified while processing node 1. In general, having processed the rings closing on nodes 1…n-1, all rings containing a node m <n can be ignored while processing node n; in other words, only the rings originating from higher numbered nodes than n are taken into account, thus operating an *upward selection*.

Obviously *full formation* requires all path-messages originating from a common source and colliding on any node of degree > 2 (nodes 4 and 6 in fig.3(*ii*)) be furtherly propagated.

Although sufficient, *full formation* of all rings is not a necessary requisite for *upward selection* since for rings that can be expressed as a linear combination of the identified elements of the SSSR being also identified from collisions on each of their node is irrelevant; referring to fig.3*(ii)-(iii)*, the even-sized ring (1,2), (2,3), (3,4), (4,7), (7,6), (6,1) in *(ii)* is obtained from the linear combination of (1,2), (2,3), (3,4), (4,5), (5,6), (6,1) and (4,5), (5,6),(6,7), (7,4) while the odd-sized one (1,2), (2,3), (3,4) ,(4,7), (7,5), (5,6), (6,1) in *(iii)* is obtained from (1,2), (2,3), (3,4), (4,5), (5,6), (6,1) and (4,5), (5,7), (7,4). Note that the inner rings (4,5), (5,6), (6,7), (7,4) and (4,5), (5,7), (7,4) in fig.3 *(ii)-(iii)* have been substituted for the shortest paths (4,5), (5,6) and (4,5), (5,7), (7,4) [4], respectively. It is worth while noticing that such paths are individuated from a collision only in the even ring case. The general result can now be stated: with regard to *upward selection* it suffices to propagate just one path-message out of all the distinct ones sharing a common origin and colliding on the same node of degree > 2 [4] - (4,5), (5,6) in the above example.

Inner-node intersection. Define an inner node-intersection of two edge-disjoint paths closing on a common node as the passage through a common node other than the first, if any, and the last. Consider now the two 5-sized, inner node-intersecting paths (1,2), (2,3), (3,4), (4,9), (9,8) and (1,5), (5,4), (4,6), (6,7), (7,8) intersecting on node 4 as in fig.4*(i)*, their union

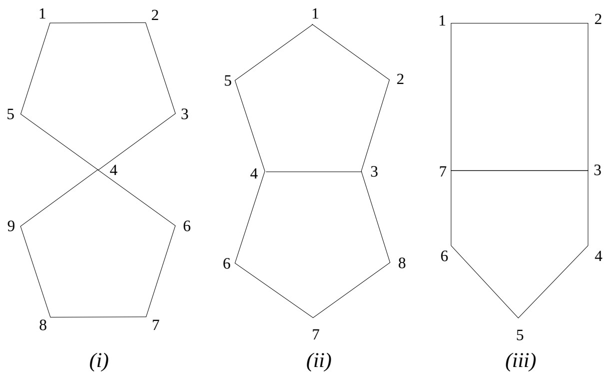

Figure 4: inner node-intersection *(i)*, inner edge intersection *(ii)-(iii)*.

is the linear combination of (and contains) two separate 5-sized rings identified and processed when considering 5-sized paths. In general, the union of any two L-sized inner-node intersecting paths originating from a common node m and closing on a node n is the linear combination of two or more shorter rings than $2L$, thus, path pairs showing an inner-node intersection can be safely ignored during the rings identification process.

Inner-edge intersection. Define an inner-edge intersection of two paths closing on a common node as the intersection on an edge other than the first one, if any, traversed in opposite directions. Consider the two 5-sized, inner-edge intersecting paths (1,2), (2,3), (3,4), (4,6), (6,7) and (1,5), (5,4), (4,3), (3,8), (8,7) intersecting on the edge (3,4) as in fig.4*(ii)*, their union, although containing two rings, represents neither a ring nor a linear combination of rings. and must not be sent the independence test. An inner-edge intersection of two paths sharing a common edge is shown in fig.4*(iii)* for paths (1,2), (2,3), (3,7), (7,6), (6,5) and (2,1), (1,7), (7,3), (3,4), (4,5). In general, the union of any two L-sized inner-edge intersecting paths originating from a common node m and closing on a node n contains two or more shorter rings than $2L$, but is neither a ring nor a linear combination of rings and must not be sent the independence test. Similar considerations apply for the union of two L-sized inner-edge intersecting paths sharing a common first edge.

Collapsed path. An edge sequence traversing a node more than once will be indicated as a collapsed path.

The union of two paths, of which at least one is

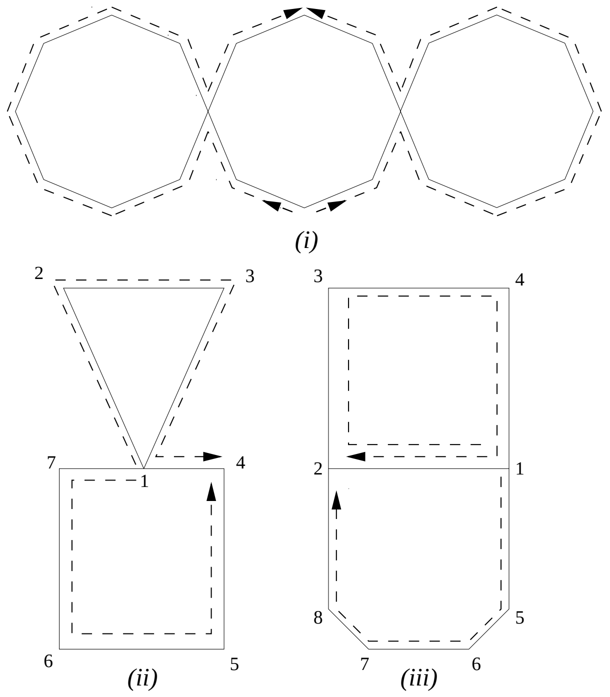

Figure 5: two collapsed path-messages union *(i)*, a collapsed path-message and a non collapsed one union *(ii)*, *(iii)*.

collapsed, sharing a common origin and closing on a common node determines an inner node-intersection and/or an inner edge-intersection (fig.5*(i)*,*(ii)*,*(iii)*). Paths collapse can easily be prevented while propagating path-messages.
The combined use of *upward selection*, pairing of paths showing no inner-node/inner-edge intersection while building rings to send the linear independence test, together with path-messages collapse preventing will be indicated as ***pre-filtering***.

5 The model

Structures. Relevant structures are now presented and described in details. The terminology used is as close as possible to the one in [1] to ease comparison wherever possible.

Adjacency matrix. The network is described by an adjacency matrix used to check the connection status of nodes during the send phase and to convert an ordered node list into an edge train and *vice versa*.

Path-messages. Each path-messages is represented by a structure made up of four fields:

1. originating node (*nfirst*)
2. last node traversed (*nlast*)
3. first edge traversed (*efirst*);
4. a dynamic integer array (DIA) tracking of all nodes traversed save *nfirst* and *nlast*.

The representation of paths as a sequence of nodes easily allows to check for paths collapse and intersection.

Binary node-encoded DIA (*bned*). A *bned* consists of an array of N bits representing a DIA in such a way that its n^{th} bit is set to 1 if the DIA contains the node n or set to 0 otherwise. Inner-node/inner-edge intersections of two paths are immediately detected by the bitwise AND of the respective *bned*'s.

Binary edge-encoded path (*beep*). A *beep* consists of on array of E bits representing a path in such a way that its n^{th} bit is set to 1 if the path contains the edge n or set to 0 otherwise. The edge representation of each path is reconstructed from a path-message through the invocation of a function, *indices_to_edge()*, operating on the couples: (*nfirst*;DIA[1]), (DIA[1];DIA[2]),. . . , (DIA[n-1];*nlast*), (*nlast;current_node*).

Binary edge-encoded ring (*beer*). A *beer* consists of on array of E bits representing a ring in such a way that its n^{th} bit is set to 1 if the ring contains the edge n or set to 0 otherwise. A *beer* is obtained from the bitwise OR of two *beep*'s.

Transceivers. Each transceiver node (*Tnode*) is represented by a structure made of a receive and a send buffer consisting in two variable length arrays of path-messages.

Collisions. According to the considerations exposed in sect.4, the collision status of two L-sized path-messages p_i and p_j simultaneously received by the same transceiver is classified as follows (fig.6):

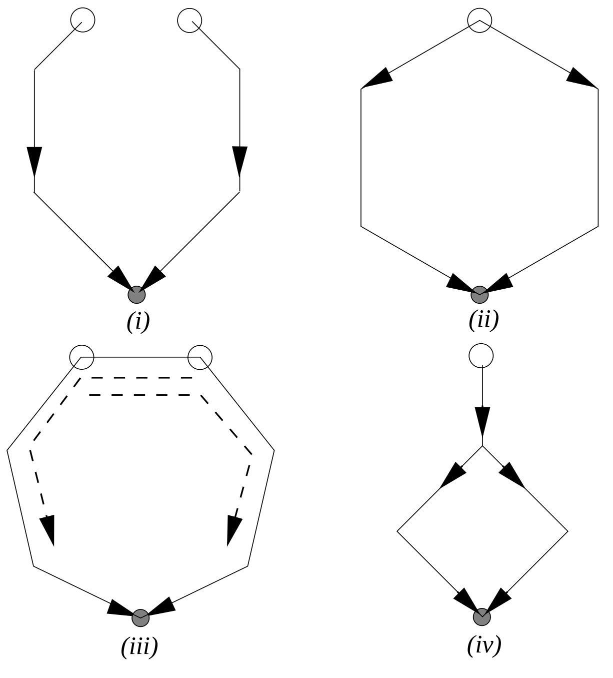

Figure 6: no collision (i), node collision (ii), inverse edge collision (iii), direct edge collision (iv).

1. no collision[1]:
   $nfirst_i \neq nfirst_j$ and $efirst_i \neq efirst_j$
   i.e., two path-messages p_i and p_j originating from distinct nodes traverse two distinct first edges. If the two path-messages are separate then their union does not form a ring, otherwise it contains shorter rings if they show an inner-node/inner-edge intersection.

2. node collision:
   $nfirst_i = nfirst_j \; efirst_i \neq efirst_j \; p_i \cap p_j = \{nfirst_i\}$,
   i.e., two path-messages p_i and p_j originating from the same node track two entirely separate paths; their union gives a ring of length $2L$;

3. inverse edge collision:
   $nfirst_i \neq nfirst_j \; efirst_i = efirst_j \; p_i \cap p_j = \{efirst_i\}$,
   i.e., two path-messages p_i and p_j originating from two distinct nodes connected by a common edge traversed in opposite directions get to a common receiver through two entirely separate paths save the first edge; their union gives a ring of length ($2L$-1);

4. direct edge collision:
   $nfirst_i = nfirst_j \; efirst_j = efirst_j$
   i.e., two path-messages p_i and p_j originating from the same node traverse their first common edge in the same direction; their union include a shorter ring than $2L$, such a ring being identified by the node collision of two paths of length $l < L$. A group of mutually direct edge collided path-messages fig.7 *(i)* can be reduced to one of its component path-message fig.7 *(ii)*[4].

Path-message functions.

***receive()*:** processes path-messages stored in the receive buffer of each *Tnode* identifying rings, submitting them to a ring selection process, moving messages to be propagated into the send buffer.

***send()*:** processes path-messages stored in the send buffer of each *Tnode* forwarding them to adjacent *Tnodes*.

[1] This definition restricts the concept of collision precedently used requiring a common originating node or edge for path-messages.

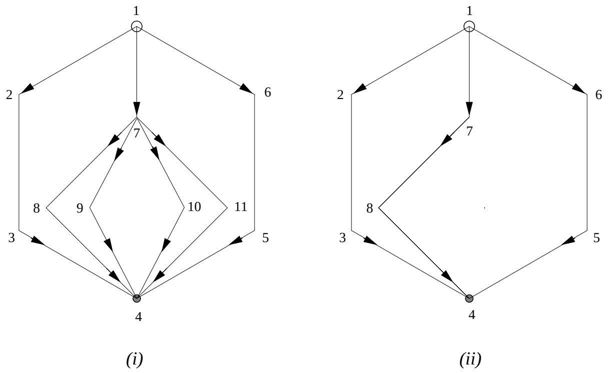

Figure 7: direct edge collided path messages *(i)*, group reduction *(ii)*.

Receive() and *send()* make use of a set of ancillary functions:

***adjacent()*:** given a pair of node indices, checks the adjacency matrix for connection and returns their connection state;

***join()*:** given a pair of node/inverse edge collided paths of length L, checks paths for passage through one of the *Tnodes* already processed and for inner-node/inner-edge intersections: if the check fails an ordered sequence of edges coded in a *beer* is returned identifying a ring of length 2L-1 or 2L to be sent the linear independence test;

***merge()*:** selects a path-message to be propagated out of a set of mutually node or direct edge collided messages;

***node_degree()*:** returns the degree of a *Tnode*;

***nodes_to_edge()*:** given a pair of node indices, returns the index of the connecting edge;

***push()*:** appends *nlast* to DIA and updates *nlast* to current sending *Tnode* index.

6 Network Initialization

The receive buffer of each *Tnode* n is set to the empty state and the send buffer contains d_n (where d_n is the *Tnode* degree) distinct path-messages of length 1. Each path-message consists of equal *nfirst* and *nlast* set to an adjacent *Tnode* n_i, *efirst* set to the edge connecting n to n_i, a zero-sized DIA. Thus the initial network state corresponds to the beginning of the second send/receive cycle.

7 Sending

The *send()* function operates on path-messages stored in the send buffers as follows:
for each $Tnode_i$ **do**
for each L-sized path-message p_{i_j} in the send buffer of $Tnode_i$ **do**
for each $Tnode_k$ adjacent to $Tnode_i$ **do**
if for p_{i_j} results $k \neq$ *nfirst,* DIA[1], ..., DIA[L-1], *nlast* **then**
copy p_{i_j} into p'_{i_j} *push()* i to p'_{i_j}
move p'_{i_j} to $Tnode_k$'s receive buffer
delete p'_{i_j}
end if
end do
delete p_{i_j}
end do
end do

8 Receiving

The *receive()* function operates on path-messages stored in the send buffers as follows (see [1] for comparison and for an efficient collision detector algorithm):
For each $Tnode_i$ $(i = 1, \ldots, N-1)$:

1. *merge()* into a single message any group of path-messages involved in a direct edge collision with each other;
2. [in the set identified by *merge()*] *join()* any pair of path-messages originating from $Tnode_{j>i}$, $Tnode_{k>i}$, involved in an inverse edge collision

and forward the resulting odd-sized ring to the ring selector;

For each *Tnode*$_i$ ($i = 1, ..., N-1$):

1. [in the set identified by merge()] *join()* any pair of path-messages originating from each $Tnode_{j>i}$ involved in a node collision and forward the resulting even-sized ring to the ring selector;

2. move any node-uncollided or direct-edge-uncollided path-message into the send buffer;

3. **If** *Tnode*$_i$ has degree> *2* **then** *merge()* into a single message any group of path-messages involved in a node collision and move the resulting path-message into the send buffer;

4. reset the receive buffer to the empty state.

9 Ring selector

Supposing $r_c <$ E-n+1 rings of the SSSR have been previously selected, the ring selector operates as follows on a submitted ring r: for i=1,....,r$_c$ if r is not a linear combination of the r$_c$ rings previously selected

1. r enters the SSSR

2. r_c is increased by 1

3. if r_c =E-N+1 then SSSR determination process is stopped.

Linear independence of rings is tested by means of a gaussian elimination procedure carried out on a bit matrix of *beer*'s maintained in upper echelon form. Operations on the matrix are carried out using modulo 2 arithmetics so that the sum (or difference) of two columns is obtained by the bitwise XOR of their elements. A copy of the *beer* to be tested is appended as a new row to the matrix and the gaussian elimination process begins to reconduce the matrix in upper echelon form. If, at any stage, the new row vanishes then the *beer* is linearly dependent and is rejected, otherwise the corresponding ring is added to the SSSR.

10 Process driving

Finally, the overall process of SSSR perception is performed as follow:

1. initialize the network

2. set the iteration counter c to one

3. increment c by one

4. call *send()*

5. call *receive()*

6. cycle over steps 3 - 5 until $E - N + 1$ linearly independent minimal rings are identified.

11 Conclusions

According to [1], the linear independence test is the most demanding part of the algorithm in terms of computational order, therefore even small changes in quantity of data to process have conspicous effects with respect to computing times. The prefiltering approach ensures no duplicated ring undergoes the linear independence test thus constituting a valid alternative to the hashing method since the prefiltering one actually meets the efficiency of a perfect hashing algorithm.

References

[1] R. Balducci and R. S. Pearlman. Efficient Exact Solution of the Ring Perception Problem, J.Chem Inf. Comput.Sci., 1994, 34, 822-831.

[2] Knut, D.E. Sorting and Searching. The Art of Computer Programming, Addison-Wesley: Reading, MA, 1973; Vol.3.

[3] Sedgewick, R. Algorithms in C. Addison-Wesley : Reading, MA, 1990

[4] Horton, J.D. A Polynomial-Time Algorithm to Find the Shortest Cycle Basis of a Graph. SIAM J. Comput. 1987, 16, 358-366.