Short communication

Extended Connectivity in Directed Graphs

Jure Zupan

National Institute of Chemistry, Ljubljana, Hajdrihova 19 Ljubljana, Slovenia

** Corresponding author: E-mail: jure.zupan@ki.si;*
phone: 00-386-(0)1-4760-279

Received: 01-02-2010

This paper is dedicated to Professor Milan Randić on the occasion of his 80th birthday

Abstract

An algorithm for the evaluation of the extended connectivity in directed graphs is described and discussed. The algorithm is a general purpose one for finding the number of all paths from any given node V_i in a directed graph toward all leaves that can be reached from that particular node V_i in the graph.

Keywords: Extended connectivity, directed graphs, algorithm, large data base.

1. Introduction

Due to the demonstrated ability of graph theory in various fields of science, including chemistry, graphs[1,2] as the representation of a certain form of interest (a molecule, an ensemble of generated structures from the given set of fragments, a network-ordered set of properties of a collection of compounds, the hierarchy of process instructions, a representation of a set of compounds employed in combinatorial synthesis, etc.) together with several simple approaches, concepts and properties of graphs have found new applications and new meanings. Together with the increased application of graph theory, the available computer power has enabled researchers to handle large graphs containing not only thousands but literally millions of nodes and connections.

Graphs can represent a population of objects of very different origin and type. Evidently, molecules (chemical structures) are easily visualized as graphs composed of atoms (as nodes) and covalent bonds between them (as links) (Figure 1a). Another type of graphs, often encountered in many fields of science or in various forms of data handling, is a directed graph. It represents a hierarchy of objects of the same kind (molecules, products, compounds, process steps, etc). A good example of a hierarchy is a collection of chemical compounds linked together into various groups (nodes) according to common structural features that are shared only by the sub-groups of the collection. Such features or properties can be, for example: aromaticity, presence of a specific atom (oxygen, phosphorus, sulfur, etc.) or specific group (carbonyl, hydroxyl, benzene ring, etc.), or any other well-defined structural feature or property. In short, any feature that is not shared by all members in the collection. In its simplest form such a collection of hierarchical properties is called a tree, when ordered according to the downward direction (Figure 1b). A directed graph has the top-most node called the root and the nodes at its end, called the leaves.

For the evaluation of graph invariants and/or other graphs' properties, any new or improved procedure that offers more efficient or optimized solution, is welcome and desired. The concept of extended connectivity in a graph is one of often used procedures in the description of graphs. In this study only connected graphs are considered.

Morgan[3] has introduced the extended connectivity as an iterative procedure applied on all nodes of the graph (representing the non-hydrogen atoms linked by the covalent bonds in the chemical structure). At the beginning of the procedure the initial values of extended connectivity coefficients ec_is at each node are the numbers of non-hydrogen atoms (nodes) linked directly to it. At each following iteration step all ec_is are replaced by the sum of the ec_is of the closest neighboring nodes (Figure 1a). In graph theoretical terms the extended connectivity at node V_i, ec_i, at the iteration step *r,* is the sum of all walks of length *r* starting at node V_i and ending at any node V_j that

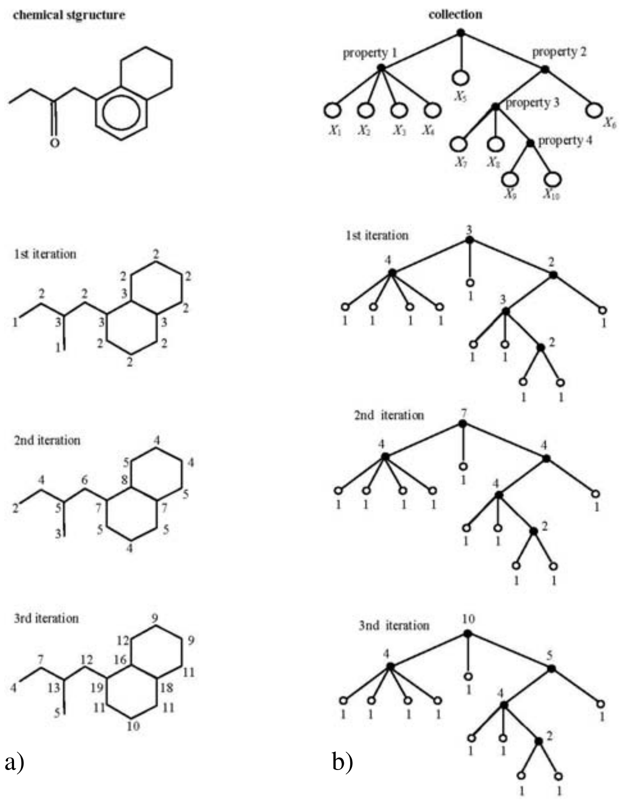

Figure 1: Ordinary and directed graphs. Ordinary graph (a) representing chemical structure composed of 15 atoms (nodes) and corresponding bonds (links), and directed graph or tree (b)**Figure 1:** Ordinary and directed graphs. Ordinary graph (a) representing chemical structure composed of 15 atoms (nodes) and corresponding bonds (links), and directed graph or tree (b) representing a collection of ten objects X_i, (i = 1,...10). The extended connectivity values ec_is after the first three iteration steps for ordinary and directed graph are shown on the left and right side, respectively. The ec_is values of the ordinary graphs (left column) are sums of *all* ec_is of neighboring nodes, while in the directed graph ec_is are sums of ec_is belonging to *downward* nodes. representing a collection of ten objects X_i, (i = 1,...10). The extended connectivity values ec_is after the first three iteration steps for ordinary and directed graph are shown on the left and right side, respectively. The ec_is values of the ordinary graphs (left column) are sums of *all* ec_is of neighboring nodes, while in the directed graph ec_is are sums of ec_is belonging to *downward* nodes.

can be reached in the particular graph[4] by the walk of length *r*.

If a graph is a directed one, the definition of the extended connectivity as used in the present work is somewhat different from the previous one. A graph is a directed one (Figure 2) if each link between two nodes V_i and V_j is associated with a specified direction 'up' or 'down'. The link (V_i , V_j), if regarded from V_i towards V_j , has opposite orientation (down or up) compared to the orientation if regarded from node and V_j towards V_i (up or down).

Additionally, if the digraph contains rings, the up and down links *within each* ring should not be cyclic, i.e., each ring must have at least one exit node mandatory accessible regard less through which entry point in the ring the particular path has been followed. This last condition provides the access to at least one leaf in any possible path in the digraph. If this condition is not met, the walks following the down direction can run in cycles never reaching any leaf. The ec_i at any node V_i is the sum of all paths that are possible from the node V_i to any *terminal* mode (leaf) of the directed graph that can be reached from that node in the *downward* direction (Figure 1b). Under the explained conditions the extended connectivity of any node V_i in a data base organized as a directed graph provides the information with how many different paths the objects (leaves) in the collection can be retrieved from any entry point (node V_i).

2. The problem

On the paper and for small graphs, the extended connectivity for each node is very simple to evaluate and assign to each of them. In the standard way for the calculation of the extended connectivity of graphs, higher order of the adjacency matrix are used[5], which, unfortunately, increases with the square of the number of nodes. When a graph contains a large number of nodes *n, n* being in order of tens of thousands or even more (large chemical structure or large collection of chemical structures), keeping track of most of the graph descriptors or graph properties becomes increasingly more complex[6]. Regardless of the representation of the graph (either as adjacency matrix or as a non-ordered list of nodes), the calculation of the extended connectivity in large graphs can be very time-consuming. This is especially true for trees (directed graphs or digraphs) representing large collections of data where individual objects or items are frequently updated – added, changed, and/or deleted, etc. what causes permanent recalculation of paths and connectivity between nodes in the graph. Therefore, updating the extended connectivity for each node in the digraph permanently under the condition of frequent update becomes a bottle-neck in the process of handling graphs.

In a tree each node has only one link oriented towards the root, while the number of downward links is completely dependent on the structure of the graph. If each node has exactly two down links, the tree is called a binary tree. If a directed graph is not a tree, but a graph containing rings (Figure 2), than at least one node has more than one upward link what enables some of the leaves to be accessed via *different* paths. As mentioned before, any ring in the digraph should not have links with directions running in cycles.

Because each node in a directed graph can have different numbers of downward and upward links, in the representation of a directed graph these two sorts of links have to be distinguished and consequently at each node the information about the orientation of each link must be provided.

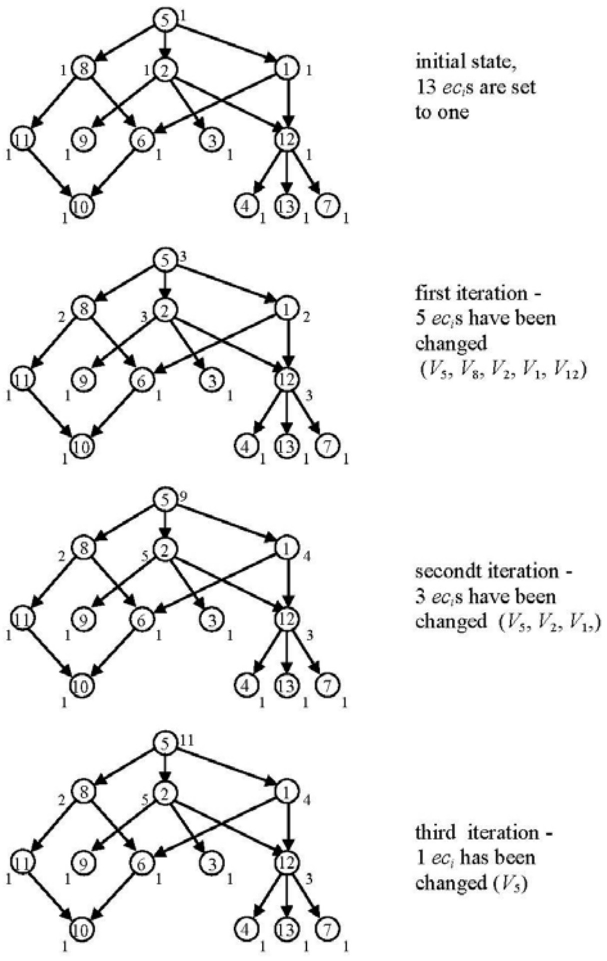
initial state, 13 ec_is are set to one
first iteration - 5 ec_is have been changed (V_5, V_8, V_2, V_1, V_{12})
secondt iteration - 3 ec_is have been changed (V_5, V_2, V_1,)
third iteration - 1 ec_i has been changed (V_5)

Figure 2: Calculation of the extended connectivity for directed graph containing four rings (closed loops). The ec_i values at each node V_i (the ID numbers of nodes are inside the circles) are calculated as a sum of ec_i values of the closest *downward* nodes from the *previous* iteration. The evaluation starts with all ec_is set to one. For example: the ec_2 value is always calculated as a sum of $ec_3 + ec_9 + ec_{12}$. In the first, in the second, and in the third iteration ec_3 has the values of 3, 5, and 5, respectively. After the third iteration all ec_i values remain unchanged.

Let us suppose that a directed graph is described by a list of *n* nodes $\{V_i\}$, $i = 1,...n$ each of which is represented by two groups of k_1 and k_2 addresses towards upward and towards downward nodes, respectively: $V_i = (\{A_{j1}^{\text{up}}\}, j_1 = 1,...k_1, \{A_{j2}^{\text{down}}\}, j_2 = 1,...k_2)$. The goal is to find the extended connectivity for the *downward* nodes in a procedure that will run efficiently on a list *n* nodes, which is linear, and not on the adjacency matrix, which grows proportional to n^2.

3. The Algorithm

The procedure is schematically shown in Figure 2. It starts with the initial ec_i value (or values) for all nodes V_i of the graph (root, nodes, and leaves). In fact this initial values do not represent the connectivity of any kind, but setting at the beginning all ec_is to one speeds up the proposed algorithm replacing two operations (checking for zero and summation) with only one summation in all cases. The procedure of calculating the actual downward extended connectivity continues iteratively. Each iteration means sequential checking for all nodes $\{V_i\}$ their down addresses $\{A_{j2}^{\text{down}}\}$ replacing the current ec_i at the node V_i with the sum of ec_is of the first down neighbors of V_i in a similar fashion the standard extended connectivity calculation is made. At each iteration the number of ec_is that have changed their values from the previous ones is recorded. When none of the ec_is have been changed the iteration stops. The obtained ec_is represent the number of all possible paths toward the nodes that can be reached from V_i.

The difference between the iterative procedure of extended connectivity calculation in ordinary graphs and directed graphs is twofold: first, in ordinary graphs at each node *all* their direct neighboring ec_is are added to obtain the new ec_i value, while in the directed graph at each node only the ec_is values of the *downward* nodes are added to yield the new ec_i, and second, the consequence of the first difference is that the iterative procedure in ordinary graphs yield larger and larger ec_is values (walks in the graph can be of any length!), while in the directed graphs *all* ec_is values are limited by the length of the longest path to the furthest leaf reachable in the downward direction from each particular node V_i.

The efficiency of the algorithm lies in the fact that the order in which the nodes are considered does not influence the results. At first glance this property does not seem to be of much value; however, in large and heavily branched data collections where most of the nodes are upward and downward multilinked, and in which the objects (leaves) are constantly updated and nodes containing (linking together) large sub-graphs often relocated, any procedure that requires either ordered nodes or recalculation of the adjacency matrix, or both. Both procedures are quadratic with the respect to n what makes them prohibitively time consuming.

The procedure of the calculation of the extended connectivity of the directed graph is discussed in detail using a small collection of six objects and six properties as an example. In Figure 2, the root representing the entire collection is the node V_5, the objects are nodes V_3,V_4, V_7, V_9, V_{12}, and V_{13}, while the nodes V_1, V_2, V_6, V_8, V_{11}, and V_{12} represent properties that each of the objects connected to a particular node has. There are four rings in the graph $\{V_5, V_2, V_{12}, V_1, V_5\}$, $\{V_5, V_8, V_6, V_1, V_5\}$, $\{V_5, V_8, V_{11}\ V_{10}, V_6, V_1, V_5\}$, and $\{V_8, V_{11}, V_{10}, V_6, V_8\}$. If object V_{10} is taken as an example, one can see that there are three (3) different paths leading to it from the root: a) V_5, V_8, V_{11}, V_{10}; b) V_5, V_8, V_6, V_{10}, and c) V_5, V_1, V_6, V_{10}.

Table 1 shows the ec_is for all 13 nodes in three consecutive iteration steps. The second column shows the sum of all *downward* nodes. The third column shows the

Table 1. Three iteration steps in the calculation of the extended connectivity values ec_i for the directed graph shown in Figure 2.

Node	Downward nodes	Initial ec_is	1st iteration	2nd iteration	3rd iteration
V_1	V_6+V_{12}	1	1+1=2	1+3=4	1+3=4
V_2	V_3+V_9+V_{12}	1	1+1+1=3	1+1+3=5	1+1+3=5
V_3	–	1	1	1	1
V_4	–	1	1	1	1
V_5	V_1+ V_2+V_8	1	2+3+1=6	4+5+2=11	4+5+2=11
V_6	V_{10}	1	1	1	1
V_7	–	1	1	1	1
V_8	V_6+V_{11}	1	1+1=2	1+1=2	1+1=2
V_9	–	1	1	1	1
V_{10}	–	1	1	1	1
V_{11}	V_{10}	1	1	1	1
V_{12}	V_4+V_7+V_{13}	1	1+1+1=3	1+1+1=3	1+1+1=3
V_{13}	–	1	1	1	1
	No. of changes in ec_is	13	5	3	none

initial values of ec_is. In the next three columns the ec_is obtained in three consecutive iteration steps are given. Each ec_is value is calculated from the ec_is values of the nodes as given by the formula in the second column. For the calculation of ec_is in the directed graphs the upward nodes (or their addresses) are not required.

For example, at the moment, the algorithm starts evaluate $ec_5 = ec_1 + ec_2 + ec_8$ (Table 1, column 4), two of the three values, ec_1 and ec_2, are already updated ($ec_1 = 2$, and $ec_2 = 3$). This yields the sum ($ec_1 + ec_2 + ec_8$) equal six (6) instead of three (3) what would follow from the initial values of ($ec_1 = ec_2 = ec_8 = 1$) if the algorithm would take their values from the *previous* iteration (initial state) and not the *updated* ones. This makes the proposed algorithm more efficient, what can be checked by comparing Figure 2 with Table 1. In Figure 2, the ec_is in each iteration are calculated separately on the basis of the values of ec_is obtained in the *previous* iteration. Therefore, it takes four iterations to achieve the stable set of ec_is values. The proposed algorithm needs only three iterations for stabilization of ec_is because it takes into account actually updated ec_is values, as shown in Table 1.

The final 'downward' extended connectivity value ec_i for any node V_i shows the number of different paths leading *downward* to all objects (leaves) from that particular node V_i. If the directed graph is a tree ec_i is equal to the number of leaves that are linked in the node. As it is shown in our example, this procedure allows the evaluation of ec_is in the directed graph even if it contains rings.

4. Conclusion

The efficiency of the algorithm for the evaluation of the extended connectivity in directed graphs depends on the length of the longest path and on the ordering of nodes along this path. The most convenient order of nodes would be the one in which the ec_is of nodes closest to the leaves will be evaluated first, and the ec_is of the nodes directly below the root last. In such a case only one iteration step is needed to calculate all ec_is even in the largest graphs. On the other hand, the slowest case of the ec_is calculation depends on the most inconvenient order in the longest path possible. Nevertheless, in heavily branched and interconnected directed graphs, the longest paths are usually very short compared to the number of all leaves and compared to the number of all nodes, hence the algorithm is very efficient. In our applications on collections consisting of more than 10,000 nodes, the algorithm needs 10–15 runs to calculate ec_is for all nodes.

It might be interesting to note that this algorithm, although invented for chemists, was applied in the dictionary of meanings of Slovenian words[7] where it works well showing all possible meaning-paths from generalized meanings (nodes) toward all single words (leaves) in the dictionary.

5. Acknowledgement

The financial support by the Agency for Research of Republic of Slovenia (ARRS) through the program grant P01–0017 and project grant J7–0382 is gratefully acknowledged.

The paper is dedicated to my friend and co-worker of many years – to Professor Milan Randiæ at the occasion of his 80-th birthday. At the same time this paper, describing an algorithm on the extended connectivity in graphs, is written as a tribute to the crucial role the late Dr. Marko Razinger has played with his research in the applicability of graph theory in chemistry through which Professor Randiæ was attracted to the collaboration with National Institute of Chemistry, Ljubljana, fifteen years ago.

6. References

1. R. B. King, *Applications of Graph Theory and Topology in Inorganic Cluster and Coordination Chemistry*, CRC Press, **1993**.
2. J. Randić, *Acta Chim. Slov.*, **1998**, *45*, 239–252.
3. H. L. Morgan, *J. Chem. Doc.*, **1965**, *5*, 107–113.
4. J. Figueras, *J. Chem. Inf. Comput., Sci.*, **1993**, *33*, 717–718.
5. M. Razinger, *Theoret. Chim. Acta (Berl.),* **1982**, *61*, 581–586.
6. M. Randić, M. Razinger*, J. Chem. Inf. Comp. Science*, **1995**, *35 (3)*, 594–606.
7. J. Zupan, *Jezik in slovstvo*, **2009**, *54 (3–4)*, 139–151.

Povzetek

Predstavljen je učinkovit algoritem za določanje razširjene stičnosti v usmerjenih grafih. Algoritem je splošen in omogoča določitev števila vseh poti od poljubnega vozla V_i v usmerjenem grafu do vseh končnih vozlov (listov), ki so dosegljivi iz omenjenega vozla V_i.