**Automatic Graph Construction of Periodic Open Tubulene ((5,6,7)3) and Computation of its Wiener, PI, and Szeged Indices**

**AHMAD YOOSOFAN** and **MOSTAFA NAMAZI–FARD**

*Department of Computer Engineering, Faculty of Engineering, University of Kashan, Kashan, Iran*

(Received March 13, 2011)

**ABSTRACT**

The mathematical properties of nano molecules are an interesting branch of nanoscience for researches nowadays. The periodic open single wall tubulene is one of the nano molecules which is built up from two caps and a distancing nanotube/neck. We discuss how to automatically construct the graph of this molecule and plot the graph by spring layout algorithm in graphviz and netwroxx packages. The similarity between the shape of this molecule and the plotted graph is a consequence of our work. Furthermore, the Wiener, Szeged and PI indices of this molecule are computed.

**Keywords:** Open tubulene, topological index, Szeged index, Wiener index, PI index.

**1 INTRODUCTION**

Carbon nanotubes exhibit a large number of new interesting phenomena, therefore many researches of different areas attracted to work on nanotubes [1]. They are crucial in all sorts of ways because of the manifold utilities they provide. One interesting feature of carbon nanotubes is their use as catalyst for improving the hydrogen absorption and desorption [2]. Some researchers are trying to use single-wall nanotubes as reservoir for storing hydrogen which may use as fuel by penetrating more Hydrogen atoms in the structure of the molecules[3,4]. One major element of energy research activities of some countries is reducing or eliminating the dependency on petroleum of transportation systems by replacing it with new fuels. Hydrogen fuel have the potential to offer cleaner, more efficient alternatives to today's technology [5]. Therefore, many fuel molecules, including
nanotubes, with different features have been found and studied up to now. Tubulenes are types of nanotubes molecules which are studied as fuel, too [3,6,7].

Some physiochemical properties of these molecules depend on their structures. Simple indirect graph is used to model the structure of these molecules. Despite its simplicity and lack of some structural characteristics of a molecule, graph of a molecule comprises important topological information. Numerical values calculated based on a molecular graph are called topological indices. Several different topological indices with various applications have been proposed up to now.

In this article, the graph of open periodic single-wall tubulene is generated by an iterative method. The graph is plotted by spring layout algorithm [8]. In addition, some topological indices of the graph of this molecule are computed.

2 Automatic Graph Construction of Periodic Open Tubulene

Periodic open single wall tubulene is one of the nano molecules which is built up from two caps and a distancing nanotube/neck [9]. Periodic closed tubulene is derived from C_{60} by cutting off polar hexagons with the repeat spherical moiety [9]. The figure of periodic closed tubulene C_{204(6(56)}^{3}(665)^{3}(656)^{3}7 − Z_{[12,0]} − r); r=4, is depicted in [10]. The periodic open tubulene is open counterpart of closed tubulene which is focused in this article. The precise figure of open periodic tubulene((5,6,7)3VA) is plotted in [10]. The Figure 1 shows a close plot of this molecule which is one of the consequences of this paper.

![Figure 1. Periodic Open Tubulene.](image)

It is difficult to construct the graph of periodic open tubulene based on its physical characterizations. Researchers who want to calculate topological indices of these sort of molecules often try to find the mathematical relations based on the graph of them [11–13]. It seems that finding the adjacency matrix of a nanotube is more useful than finding only one or two indices of it. One of the most interesting advantages of constructing graph of a
molecule is the straightforward calculation of the various topological indices. A novel algorithm for constructing the graph of carbon nanohorn is proposed in [14]. In this article, an analogous approach has been proposed for constructing the graph of periodic open tubulene which is discussed in detail in the following paragraphs.

The repetition of three connected sections of pentagons, hexagons and heptagons constructs the graph of the open tubulene. These sections are shown in the Figure 1 by blue, green and red colors, respectively. The first and the second sections connect to each other to construct levels of this molecule. The third section uses for conjunction of the current level to the previous level, if any exist. Based on this definition of levels, the Figure 1 has three levels, three first sections, three second sections and two third sections. Each section rolls different number of pentagons, hexagons and heptagons.

The first section consists of six hexagons and three pentagons, Figure 2. Each consecutive pair of hexagons interleaves by a pentagon. The Figure 3 has more details on construction of the first section of the first level. It shows the node number of this section and how a node connects to other nodes, too. The construction of the first sections of the subsequent levels is similar to the first section of the first level.

Figure 2. The First Section.
The second section also consists of six hexagons and three pentagons with different arrangement in respect to the first section. The second section is connected to the first one which is shown in the Figure 4.
The third section is too simple. It consists of six connected heptagons. The Figure 5 shows the third section.

Figure 5. The Third Section.

Suppose NL denotes the number of levels has been created up to now, and ML denotes the maximum number of levels which we need to construct the graph. The algorithm of constructing open periodic tubulene has the following steps:

1. Construction of the first section of the first level
2. Construction of the second section of the first level
3. NL = 1
4. WHILE NL < ML DO
   4.1. Construction of the first section of the (NL+1)th level
   4.2. Construction of the third section of the (NL+1)th level
   4.3. Construction of second section of the (NL+1)th level
   4.4. NL = NL + 1
5. END

The Python programming language is used to implement the algorithms discussed in this paper. Python is a powerful open source and free scripting language enriched by many open source modules for wide variety of purposes. It has a very concise, clear, readable and consistent syntax, yet it has a lot of capabilities and advanced features such as dynamic typing, generators, exceptions, very high-level dynamic data types and classes. Several open source and free libraries are developed for working with graphs in Python,
such as python−graph, NetworkX, py_graph, graph−tool, igraph, etc. The NetworkX is used for creating and manipulating graph objects in this paper. Many types of graphs, including simple graphs, directed graphs, and graphs with parallel edges and self-loops are implemented in NetworkX [15]. The following code is the python implementation of the preceding algorithm for creating open tubulene. The sections and levels are connected together by pre, nextin, nextst. Variable nv contains the number of nodes created up to now.

\[
\begin{align*}
\text{Pre} & = [0, 0, 0, 0, 0, 0] \\
\text{nextin} & = [0, 0, 0, 0, 0, 0, 0, 0, 0] \\
\text{nextst} & = [0, 0, 0, 0, 0, 0] \\
\text{nv} & = 0 \\
\text{nv} & = \text{FirstSection}(\text{nv}, \text{pre}, \text{nextin}) \\
\text{nv} & = \text{SecondSection}(\text{nv}, \text{nextin}, \text{nextst}) \\
\text{for} & \text{ i in range}(0, \text{numberOfLevels} - 1): \\
\text{nv} & = \text{FirstSection}(\text{nv}, \text{pre}, \text{nextin}) \\
\text{ThirdSection}(\text{nextst}, \text{pre}) \\
\text{nv} & = \text{SecondSection}(\text{nv}, \text{nextin}, \text{nextst})
\end{align*}
\]

3 DRAWING THE GRAPH

During development of the preceding algorithm, plotting the resulting graph as shown in Figures 4 and 5 was used to adjust the algorithm. Therefore, several free and open source graph drawing tools were tried out. Neato was selected for graph drawing because the plotted graph by Neato is close to the shape of the molecule. Neato is a part of Graphviz package that make layouts of undirected graphs. Graphviz is free and open source graph visualization which is widely used in many areas. The tools in Graphviz take description of a graph in a simple text language, and create diagrams in different formats, such as images and SVG for web pages, PDF or Postscript for inclusion in other documents. Many useful features for concrete diagrams have been added to Graphviz, such as options for colors, fonts, tabular node layouts, line styles, hyperlinks, roll, and custom shapes.

Graphviz offers both graphical and command line tools. There exist several ways for using Graphviz from python, but in most cases the Graphviz command line tools are called to parse files containing a graph definition and render a rasterized image of the graph. Therefore, Neato can be either run in command line, or invoking it in python by “os.system” function. This is unsatisfactory for our purposes, and a more direct interface to the layout algorithms is desirable. There are several python interface libraries to the Graphviz (e.g. PyGraphviz, pydot, etc.). PyGraphviz have been chosen for this purpose in this research.

Neato draws undirected graph using a variation of spring algorithm proposed by Kamada and Kawai [8]. The proposed algorithm places an ideal spring between every pair
of nodes such that its length is set to the shortest path distance between the endpoints. The first spring algorithm was proposed by Eades [16]. Since then several researchers have proposed variations of the spring algorithm [17–19]. These algorithms are also known as multidimensional scaling, in statistics. Kruskal and Seery noted their application to graph drawing in the late 1970s [19]. Spring algorithms are the most simplest and popular algorithms in force-directed placement graph drawing methods. The graph drawing problem is modeled by a force-directed algorithm models through a physical system of bodies with forces acting between them which minimizing the energy of the system by finding a good placement of the bodies [20]. Force-directed based graph drawing algorithms yield reasonable layouts in respect to symmetry, structure, clustering and vertex distribution [21].

The plotted graph of this molecule with 3 levels is shown in the Figure 1. The plotted graph by the spring algorithm is surprisingly close to the shape of this molecule and proves the usefulness of this algorithm for drawing real molecule as previously noted in [14]. Neato supports several shapes for nodes of a graph including circle, point, etc. The shape of nodes of Figures 3 is circle, but point used in Figures 1, 2, 4 and 5 for achieving the better figure. Neato also supports more dimensions for drawing a graph. The Figure 6 shows the three dimensional image of plotted graph of the molecule with four levels. It is difficult to realized the sections and levels in a two dimensional figure which holds a three dimensional plotted graph.

Figure 6. 3D View of Open Periodic Tubulene Plotted by Neato.
4 TOPOLOGICAL INDICES

A numerical invariant related to molecular graph of a chemical compound is called a topological index [22, 23]. Also there is a semi-empirical index which is discussed in [24,25]. Several different topological indices have been proposed to encode chemical properties of molecules [26]. These indices are calculated based on graphs of molecules or graphs of different kinds of networks such as social network [27]. So, by finding the graph of a molecule (or its adjacency matrix), computing the topological indices of that molecule is straightforward.

4.1 THE WIENER INDEX

The first topological index which is used in chemistry is the Wiener index. Harold Wiener developed and used the Wiener index to determine physicochemical properties of types of alkanes known as paraffin in 1947 [28]. To define, we assume that G is an indirect simple graph. The Wiener index, W(G), of G with n vertices is the sum of the lengths of shortest paths between all pairs of vertices of G.

\[ W(G) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij}, d_{ii} = 0. \]  

4.2 SZEGED INDEX

Ivan Gutman introduced the Szeged index in 1994 [29]. Let E(G) be the set of all edges of G and e = uv be an edge of G. Define \( W(e) = n_u(e) \times n_v(e) \), where \( n_u(e) \) is the number of vertices of G closer to u than v, and \( n_v(e) \) is defined analogously. The Szeged index of G is the sum of \( W(e) \) over all edges of G [12]. So,

\[ S_Z(G) = \sum_{e \in E(G)} W(e) \]  

4.3 PI INDEX

The PI index was introduced by P. V. Khadikar in 2000 [30]. The summation over all edges uv of G which are not equidistant to u and v is PI index [31]. Based on the notations introduced in Szeged index, the PI index is defined as follows:

\[ PI(G) = \sum_{e \in E(G)} [n_u(e) + n_v(e)]. \]
5 RESULTS

Finding the shortest paths between all nodes in a graph is the most time consuming part of computing the preceding topological indices. The well known Dijkstra algorithm finds these shortest paths, but it is impossible to compute these topological indices based on this algorithm in a reasonable time. Therefore, the Floyd-Warshal algorithm is more suitable for this problem. This algorithm is basically equivalent to the transitive closure algorithm independently proposed by Roy [32] in 1959. Current version of this algorithm was proposed by Ingerman which used three nested for-loop. This algorithm is faster at the expense of memory. Therefore, the pitfall of this algorithm is the order of memory usage. The preceding results are computed on a computer with 12 GB main memory running Ubuntu 64-bit. The Table 1 shows the Wiener index, PI index and Szeged index of open tubulene with different level numbers.

Table 1: Values of some computed topological indices of different open Tubulene with different number of vertex

<table>
<thead>
<tr>
<th>Level Number</th>
<th>Number of Vertices</th>
<th>Wiener Index</th>
<th>PI Index</th>
<th>Szeged Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>48</td>
<td>5496</td>
<td>2652</td>
<td>26622</td>
</tr>
<tr>
<td>10</td>
<td>480</td>
<td>2373096</td>
<td>303108</td>
<td>22289022</td>
</tr>
<tr>
<td>20</td>
<td>960</td>
<td>18575496</td>
<td>1238388</td>
<td>187132182</td>
</tr>
<tr>
<td>30</td>
<td>1440</td>
<td>62425896</td>
<td>2807268</td>
<td>646311342</td>
</tr>
<tr>
<td>40</td>
<td>1920</td>
<td>147748296</td>
<td>5009748</td>
<td>1551890502</td>
</tr>
<tr>
<td>50</td>
<td>2400</td>
<td>288366696</td>
<td>7845828</td>
<td>3055933662</td>
</tr>
<tr>
<td>70</td>
<td>3360</td>
<td>790787496</td>
<td>15418788</td>
<td>8467667982</td>
</tr>
<tr>
<td>100</td>
<td>4800</td>
<td>2304738696</td>
<td>31530228</td>
<td>24875349462</td>
</tr>
<tr>
<td>120</td>
<td>5760</td>
<td>3982199496</td>
<td>45439188</td>
<td>43114603782</td>
</tr>
<tr>
<td>140</td>
<td>6720</td>
<td>6323212296</td>
<td>61882548</td>
<td>68613762102</td>
</tr>
<tr>
<td>160</td>
<td>7680</td>
<td>9438369096</td>
<td>80860308</td>
<td>102589336422</td>
</tr>
<tr>
<td>180</td>
<td>8640</td>
<td>13438261896</td>
<td>102372468</td>
<td>146257838742</td>
</tr>
<tr>
<td>200</td>
<td>9600</td>
<td>18433482696</td>
<td>126419028</td>
<td>200835781062</td>
</tr>
</tbody>
</table>

The plot of these indices versus vertex numbers is more readable and clear than numbers in Table 1. The Matplotlib visualization library is a standard package for curve plotting in Python. The Figures 7, 8 and 9 have been obtained using this library. The Matplotlib
automatically regulates the axes ratio. It shows the axes ratio on the plot. The diagram of the Wiener index versus vertex number is shown in Figure 7. The axes ratio of this plot is $10^{10}$.

![Figure 7. The Wiener Index Versus Vertex.](image)

The diagram of the Szeged index versus vertex number is shown in Figure 8.

![Figure 8. The Szeged Index Versus Vertex.](image)
The diagram of the PI index versus vertex number is shown in Figure 9.

**Figure 9.** The PI Index Versus Vertex.

<table>
<thead>
<tr>
<th>n</th>
<th>RMS</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1886833697</td>
<td>−5214293688</td>
<td>1965227</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>262437466</td>
<td>1771561032</td>
<td>−1487965</td>
<td>328.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1.69458e−06</td>
<td>−5304</td>
<td>155</td>
<td>−7e−13</td>
<td>0.02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1.8528e−06</td>
<td>−5304</td>
<td>154.99</td>
<td>4e−12</td>
<td>0.02</td>
<td>2e−20</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1.026e−05</td>
<td>−5304</td>
<td>155</td>
<td>−7e−13</td>
<td>0.02</td>
<td>−1e−20</td>
<td>5e−25</td>
</tr>
</tbody>
</table>

**Table 2:** Polynomial for the Wiener Index.

<table>
<thead>
<tr>
<th>n</th>
<th>RMS</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7820939</td>
<td>−29239212</td>
<td>14422.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1.04e−08</td>
<td>1428</td>
<td>−31.5</td>
<td>1.375</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3.22e−08</td>
<td>1428</td>
<td>−31.5</td>
<td>1.375</td>
<td>3.4e−19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2.27e−08</td>
<td>1428</td>
<td>−31.5</td>
<td>1.375</td>
<td>−1e−18</td>
<td>6e−23</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>9.4e−08</td>
<td>1427.99</td>
<td>−31.5</td>
<td>1.375</td>
<td>−2e−18</td>
<td>2.e−22</td>
<td></td>
</tr>
</tbody>
</table>
### Table 4: Polynomial for The Szeged Index.

<table>
<thead>
<tr>
<th>n</th>
<th>RMS</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20635483310</td>
<td>−56905553562</td>
<td>21396835.25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2886812122</td>
<td>19486947558</td>
<td>−16364897</td>
<td>3592.25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2.802e−05</td>
<td>−282138</td>
<td>4423.25</td>
<td>−21.25</td>
<td>0.22917</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1.66e−05</td>
<td>−282137.9</td>
<td>4423.25</td>
<td>−21.25</td>
<td>0.22917</td>
<td>2e−19</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>9.0298e−05</td>
<td>−282138</td>
<td>4423.25</td>
<td>−21.25</td>
<td>0.22917</td>
<td>8.7e−19</td>
<td>−3e−23</td>
</tr>
</tbody>
</table>

In Tables 2–4, RMS denotes the root mean square error of curve fitting. The polynomials recorded in Tables 2–4, when \( n = 3, 2, 3 \), are the best polynomials that is fitted to the Wiener, PI and Szeged indices of this molecule, respectively.

## 6 Conclusions

A new intuitive method for constructing the graph of open tubulene is proposed and discussed in this article. Several packages and tools based on Python programming language are used to implement the algorithm. The spring method is used to plot the constructed graph. A consequence of using this method is the similarity between the picture of open periodic tubulene and plotted graph. Three major topological indices, namely the Wiener, PI and Szeged indices of this molecule are calculated based on the constructed graph. Memory and time cost is the problems against calculating these indices unbounded.

**Acknowledgement.** This research has been supported by the research affair of the University of Kashan, I R Iran. I am very pleased from professor A R Ashrafi for helping me to correct the first draft of this paper.

### References


