# A Linear Algorithm for the Hyper-Wiener Index of Chemical Trees

Roberto Aringhieri

Dipartimento di Informatica, Università di Pisa, Corso Italia 40, 56125 Pisa, Italy

Pierre Hansen*

GERAD - École des Hautes Études Commerciales, 3000, chemin de la Côte-Sainte-Catherine, Montréal, Canada, H3T 2A7

Federico Malucelli

Dipartimento di Elettronica, Politecnico di Milano, Via Ponzio 34/5 20133 Milano, Italy

Received November 7, 2000

An algorithm with a complexity linear in the number of vertices is proposed for the computation of the Hyper-Wiener index of chemical trees. This complexity is the best possible. Computational experience for alkanes is reported.

## 1. INTRODUCTION

The Hyper-Wiener index (*WW*) is a generalization, due to Randić,[1] of the much studied Wiener index (*W*) or graph invariant.[2−4] Recall that W is defined as the sum of distances between pairs of vertices of the graph under study. Let $T = (V,E)$ denote a tree, i.e., a connected and acyclic graph. Let $e = \{i, j\}$ denote an edge of $T$ connecting the adjacent vertices $i$ and $j$. Removal of $e$ from $T$ gives two subtrees $T_i = (V_i,E_i)$ and $T_j = (V_j,E_j)$ with $|V_i|$ and $|V_j|$ vertices, respectively, such that $1 \leq |V_i| \leq n - 1$, $1 \leq |V_j| \leq n - 1$ and $|V_i| + |V_j| = n$ ($n = |V|$). Then, as already noted by Wiener

$$W(T) = \sum_e |V_i||V_j| \tag{1}$$

where the sum is over all edges of $T$. This leads to an algorithm for $W$ which has an $O(n)$, i.e., linear and the best possible, complexity.

Randić generalizes $W$ to $WW$ by considering paths $p$ instead of edges $e$ in (1): let $i$ and $j$ denote the endpoints of a path $p$ in $T$. Removing all edges of $p$ from $T$ yields a forest, i.e., a set of subtrees of $T$, possibly reduced to single points. Let $T_i$ and $T_j$ denote the subtrees (or trees for short) containing vertices $i$ and $j$, respectively. Let $p = p_{ij}$ denote the path that has endpoints $i$ and $j$. Then, by definition

$$WW(T) = \sum_p |V_i||V_j| \tag{2}$$

where the sum is over all paths of $T$.

The purpose of the present paper[5] is to show that (2) can be exploited as (1), but in a more complex way, to obtain an $O(n)$ algorithm for $WW$ in chemical trees, i.e., trees with a maximum degree less than or equal to 4. Extension to general trees is straightforward. Computational experience for alkanes with up to 25 vertices is also reported.

Before turning to this new algorithm we briefly review literature on *WW*.

The Hyper-Wiener index was defined by Randić using the Wiener matrix.[6,7] This matrix contains the values $|V_i||V_j|$ as defined above as its $(i,j)^{th}$ components and is a useful source of graph invariants.

Klein et al.[8] provide an interesting interpretation of $WW(T)$: they show that

$$WW(T) = \frac{1}{2}\Big[\sum_{i<j} d_{ij}^2 + \sum_{i<j} d_{ij}\Big] \tag{3}$$

$$= \frac{1}{2}\sum_{i<j} d_{ij}^2 + \frac{1}{2}W(T) \tag{4}$$

Thus the Hyper-Wiener index is proportional to the Wiener index plus the sum of squared distances between pairs of vertices. Gutman et al.[9] study general trees with extremal WW index. Their main result is the following

$$WW(S_n) \leq WW(T) \leq WW(P_n) \tag{5}$$

where $S_n$ and $P_n$ denote respectively the star and the path or chain with $n$ vertices. Moreover, if $T \neq S_n$, $T \neq P_n$, and $n \geq 5$

$$WW(S_n) < WW(T) < WW(P_n) \tag{6}$$

Hyper-Wiener indices of particular families of trees have been studied by several authors.[10−12] Only one paper[13] presents a detailed description of an algorithm for computing $WW(T)$ and its complexity is in $O(n^2)$.

Four extensions of the concept of Hyper-Wiener index to cyclic graph have been proposed by three groups of authors.[8,14−16]

* Corresponding author phone: (514)340-6053 ext. 5675; e-mail: pierreh@crt.umontreal.ca.

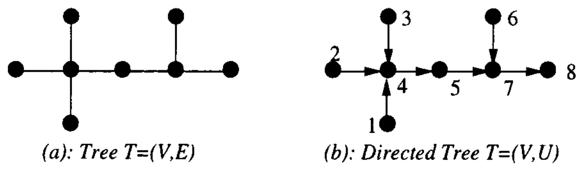

**Figure 1.** Good numeration of a tree.

Comparisons with other indices for the cyclic and acyclic cases are numerous.[16−21]

## 2. A NEW ALGORITHM FOR *WW*

The algorithm is based upon the following reformulation of eq 2:

$$WW = \sum_p |V_i||V_j| = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} |V_i||V_j| \tag{7}$$

$$= \sum_{i=1}^{n-1} |V_i| \sum_{j=i+1}^{n} |V_j| \tag{8}$$

or setting

$$M_i = \sum_{j=i+1}^{n} |V_j| \tag{9}$$

$$WW = \sum_{i=1}^{n-1} |V_i| M_i \tag{10}$$

The linearity of the computation of *WW* depends on the fact that, for a given $i$, $M_i$ can be computed recursively in constant time, knowing the values of the $M_j$, $j > i$. Equation 7 expresses that all paths in $T$ can be enumerated without repetition by considering all possible initial vertices $i$ and terminal vertices $j$ eq 8 exploits the fact that for a given $i$ the number of vertices in the subtree containing $i$ obtained when removing the path $p_{ij}$ is independent of $j$. There is a slight abuse of notation in these equations, as the $|V_i|$, $|V_j|$ depend on the path from $i$ to $j$. This will he taken care of below.

A first step is to compute values of $|V_i|$ and $|V_j|$ for all edges, as done when computing $W$. To this effect, we renumber the vertices and orient the edges (i.e., transform them into arcs) in such a way that in the directed tree $T = (V,U)$ obtained there is always an oriented path from any vertex $i$ to vertex $n$ and

$$(i,j) \in U \Rightarrow i < j$$

i.e., any arc goes from a vertex with a lower index to a vertex with a higher one. The resulting tree is a *reversed arborescence*.[22] Such a numbering of vertices is called a *good numeration* and is always possible for trees.[23] An easy way to obtain a good numeration of the vertices of $T$ is to set $k = n$ and perform a depth-first search of the tree,[24] beginning at a pendant vertex and each time an unlabeled vertex is reached assigning it label $k$ and reducing $k$ by 1.

**Example 1.** Consider the tree of Figure 1a with 8 vertices, initially indexed in any way. Applying the good numeration procedure yields the indexed directed tree of Figure 1b.

**Table 1.** Computation of the $|V_i|$, $|V_i'|$ and $W$

| $i$ | $\|V_i\|$ | $\|V_i'\|$ | $\|V_i\| \times \|V_i'\|$ |
|---|---|---|---|
| 1 | 1 | 7 | 7 |
| 2 | 1 | 7 | 7 |
| 3 | 1 | 7 | 7 |
| 4 | 4 | 4 | 16 |
| 5 | 5 | 3 | 15 |
| 6 | 1 | 7 | 7 |
| 7 | 7 | 1 | 7 |
| | | | $W = 66$ |

Rules followed for good numeration are given next.

**Algorithm CN.** 1. Given a tree $T$, described by lists of neighbors of each vertex, select a pendant vertex, label it $n$, and set $k = n - 1$;

2. If the currently visited vertex has one or several unlabeled neighbors, choose one and go there, label it $k$, and reduce $k$ by 1;

3. If the currently visited vertex has no unlabeled neighbor, backtrack to the neighbor from which it was first reached; if this vertex is the first one considered (labeled $n$) stop; otherwise return to 2.

An optional tie-breaking rule is to choose first the unlabeled neighbor of the currently visited vertex with smallest degree (as done in Example 1). An alternate stopping rule would be to interrupt the algorithm as soon as a vertex has received label 1.

A more precise, Pascal-like description of the algorithm good numeration is given in Box 1.

```
Box 1: Good Numeration algorithm:
Input: T=(V,E); Output: Label;
begin
  Let S be an empty stack;
  for each u in V do Visited(u):=false;
  Let u s.t. Degree(u)=1;
  Label(u):=n; Next:=n-1; Visited(u):=true;
  for each v in Adjacent(u) do Push(v,S);
  while S is not empty do
  begin
    u:=Pop(S); Label(u):=Next; Next:=Next-1;
    Visited(u):=true;
    for each v in Adjacent(u) s.t. not Visited(u)
    do Push(v,S);
  end;
end.
```

Let us now consider any arc $(i,j)$ of $T$; to avoid ambiguity we note the cardinality of the second subtree obtained after its removal by $|V_i'|$. Each arc of $T$ can be viewed as indexed by the index of its initial vertex. So the $i$ in $|V_i'|$ then refers to the $i$th arc.

The following recursive formula allows computation of the $|V_i|$ and $|V_i'|$:

$$|V_i| = 1 + \sum_{k:(k,i)\in U} |V_k| \tag{11}$$

$$|V_i'| = n - |V_i| \tag{12}$$

Equation 11 expresses that $|V_i|$ is equal to 1 plus the cardinalities $|V_k|$ of the (at most 3) subtrees obtained by deletion of an arc incident with $i$ which is not the $i$th one.

**Example 1 (Continued).** Applying eqs 11 and 12 for $i = 1,2,...,7$ to the tree of Figure 1b gives the results of Table 1, and the value $W = 66$ of its Wiener index.

Table 2. Computation of the $N_i$

| $i$ | $N_i$ | $i$ | $N_i$ | $i$ | $N_i$ |
|---|---|---|---|---|---|
| 1 | 1 | 4 | 7 | 6 | 1 |
| 2 | 1 | 5 | 12 | 7 | 16 |
| 3 | 1 | | | | |

Edge-labels computations can be done by applying good numeration and then formulas (11) and (12) in order of increasing indices of $i$. However, one may observe that $|V_i|$ and $|V'_i|$ do not depend on orientation of arcs. An alternate and slightly more general way of performing edge-labels computations is to consider the undirected tree $T = (V,E)$ and computes the $|V_i|$ and $|V'_i|$ recursively for pendant vertices, removing pendant edges. Rules are then the following.

**Algorithm EL.** 1. Given a tree $T = (V,E)$ give to all its vertices $i$ a value $t_i = 1$;

2. Consider all pendant vertices of $T$; for each such vertex: set $|V_i| = t_i$, add $t_i$ to the value $t_j$ of the adjacent vertex $j$ of $i$, and remove edge $(i,j)$ from $E$;

3. If $E$ is nonempty return to 2; otherwise stop.

A Pascal-like description of the algorithm edge-labels computation which computes the $|V_i|$ and $|V'_i|$ is given in Box 2.

```
Box 2: Edge Labels Computation algorithm:
Input: T=(V,E); Output: |V(i)| and |V'(i)| for (i,j) in E.
begin
   for each i in V do t(i):=1;
   repeat
      for each (i,j) in E s.t. Degree(i)=1 do
      begin
         |V(i)|:=t(i); |V'(i)|:=n-t(i);
         E:=E \ {(i,j)};
         t(j):=t(j)+|V(i)|
      end
   until E is empty
end.
```

Another preliminary computation is the sum $N_i$ of the $|V_j|$ for all vertices $j$ in a subtree $T_i$. It is performed with the following recursive formula

$$N_i = 1 + \sum_{k:(k,i)\in U} (|V_k| + N_k) \qquad (13)$$

In this equation

$$1 + \sum_{k:(k,i)\in U} |V_k| = |V_i| \qquad (14)$$

and is added to the sum of $|V_j|$ for all subtrees $T_k$ obtained by deletion of an arc incident with $i$ and not the $i$th one.

**Example 1 (Continued).** Applying eq 13 for $i = 1,2,...,7$ to the tree of Figure 1b gives the results of Table 2.

We can now describe the recursion for the computation of the $M_i$, which is central to our algorithm. The general case is illustrated in Figure 2.

Assume the $M_j$ have been computed for all $j > i$ and one wishes to compute $M_i$, where $i$ is the initial vertex of arc $(i,l)$; there are at most two subtrees $T_k$ and $T_{k'}$ such that $i \neq k,k'$, $(k,l) \in U$, and $(k',l) \in U$, and one subtree $T_q$ such that $(l,q) \in U$.

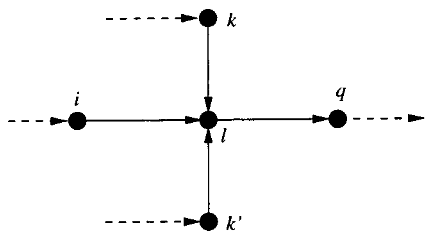

Figure 2. General case for computation of $M_i$.

Table 3. Computation of the $M_i$ and $WW$

| $i$ | $l$ | $k$ | $M_l$ | $\|V'_i\|$ | $N_k$ | $M_i$ | $\|V_i\|$ | $\|V_i \times M_i\|$ |
|---|---|---|---|---|---|---|---|---|
| 7 | 8 | - | 0 | 1 | 0 | 1 | 7 | 7 |
| 6 | 7 | - | 1 | 7 | 0 | 8 | 1 | 8 |
| 5 | 7 | 6 | 1 | 3 | 1 | 5 | 5 | 25 |
| 4 | 5 | - | 5 | 4 | 0 | 9 | 4 | 36 |
| 3 | 4 | - | 9 | 7 | 0 | 16 | 1 | 16 |
| 2 | 4 | 3 | 9 | 7 | 1 | 17 | 1 | 17 |
| 1 | 4 | 2,3 | 9 | 7 | 1,1 | 18 | 1 | 18 |
| | | | | | | | | $WW = 127$ |

A path $p_{ij}$ beginning at $i$ may either description

(i) end in $l$; then $|V_j| = |V'_i|$;

(ii) end within the subtree $T_q$; then $|V_j|$ is taken into account in $M_l$;

(iii) end within $T_k$ (or $T_{k'}$); then $|V_j|$ is taken into account in $N_k$ (or $N_{k'}$).

Note that, due to good numeration, case (iii) is only possible if $k > i$ (or $k' > i$), which implies $j > i$ for all $j \in V_k$ (or $j \in V_{k'}$).

We therefore have the recursive formula:

$$M_i = M_l + |V'_i| + \sum_{k:(k,l)\in U, k>i} N_k \qquad (15)$$

**Example 1 (End).** Applying eq 15 for $i = 7, 6, ..., 1$ to the tree of Figure 1b gives the results of Table 3 and the value $WW = 127$ of the Hyper-Wiener index.

To summarize, the new algorithm for the Hyper-Wiener index consists of the following steps:

**Algorithm LWW.** 1. Apply good numeration to label the vertices of $T$ and orient its edges;

2. Compute $|V_i|$ and $|V'_i|$ for $i = 1, 2, ..., n - 1$ by eqs 11 and 12;

3. Compute $N_i$ for $i = 1, 2, ..., n - 1$ by eq 13;

4. Compute $M_i$ for $i = n - 1, ..., 1$ by eq 15;

5. Compute $WW$ by eq 10.

**Theorem 1.** *Algorithm LWW computes the Hyper-Wiener index WW(T) of a chemical tree T in O(n) time.*

**Proof.** Correctness of the algorithm follows from the reformulation (7)−(10) of the definition of $WW$ and justifications of the various recursions given above. The $O(n)$ complexity is due to the fact that depth-first search is linear in $n$, the recursions of eqs 11, 13, and 15 take constant time per application, i.e., $O(n)$ in all, as does eq 10. A lower bound of $\Omega(n)$ is due to the size of the input. So algorithm LWW is $O(n)$ and is the best possible.

A Pascal-like description of the algorithm LWW is given in Box 3.

**Table 4.** Computing Time in msec

| *n* | codes | time | av time | % for *WW* comput. |
|---|---|---|---|---|
| 14 | 1858 | 40 | 0.0215 | 25.0 |
| 15 | 4347 | 110 | 0.0230 | 20.0 |
| 16 | 10359 | 240 | 0.0232 | 20.8 |
| 17 | 24894 | 640 | 0.0261 | 24.6 |
| 18 | 60523 | 1670 | 0.0274 | 26.8 |
| 19 | 148284 | 4250 | 0.0283 | 24.8 |
| 20 | 366319 | 11070 | 0.0304 | 23.4 |
| 21 | 910726 | 29250 | 0.0322 | 24.5 |
| 22 | 2278658 | 77010 | 0.0337 | 24.3 |
| 23 | 5731580 | 202500 | 0.0353 | 24.6 |
| 24 | 14490245 | 533900 | 0.0387 | 24.9 |
| 25 | 36797588 | 1421100 | 0.0387 | 24.8 |

```
Box 3: LWW algorithm:
Input: T=(V,E); Output: WW.
begin
   for each i in V do
   begin
      M(i):=0; Maux(i):=0; visited(i):=false
   end;
   curr:=n-1; precnod:=n; visited(n):=true;
   visited(n-1):=true;M(curr):=1;
   while (curr not equal to 1) do
   begin
      if {u in Adjacent(curr):
            visited(u)=false and u<curr} is not empty
      then begin
         prec:=curr; curr:=u;
         M(curr):=M(prec)+Maux(prec)+V'(curr)
      end else begin
         temp:=curr; curr:=prec; prec:=temp;
         Maux(curr):=Maux(curr)+Maux(prec)+V(curr)
      end
   end;
   WW:=0;
   for i:= 1 to n-1 do WW:=WW + ( V(i) M(i) );
end.
```

**Remark 1.** As mentioned above, algorithm LWW is readily extended to the case of general trees *T*. Theorem 1 still holds, i.e., the complexity is still $O(n)$. To show this, the proof must be amended by using amortized complexity arguments: while the computation of one value of $|V_i|$, $N_i$ or $M_i$ by eqs 11, 13, or 15 can then take $O(n)$ time, the time required to compute them for all *i* is in $O(n)$, and not $O(n^2)$, as it is in all three cases proportional to the number of edges, and hence of vertices, of *T*.

## 3. COMPUTATIONAL RESULTS

Algorithm LWW was coded in C$^{++}$ and applied to the computation of *WW* for all chemical trees with $n \leq 25$. The results were obtained by using a file of numerical *N*-tuple codes[26,27] obtained following the rules described in Kvasnicka and Pospichal[29] and implemented and tested by Aringhieri.[28] Actually, the reported computational results are obtained on a Linux Pentium III 600 MHz computer with 128 MB RAM memory.

Computing times are presented in Table 4. Total computing times augmented rapidly as the number of alkanes grows exponentially with *n*. Average computing times per molecule are represented in Figure 3 and are clearly linear in *n*. Using linear regression we obtain

$$t = -0.00124848 + 0.00158881n$$

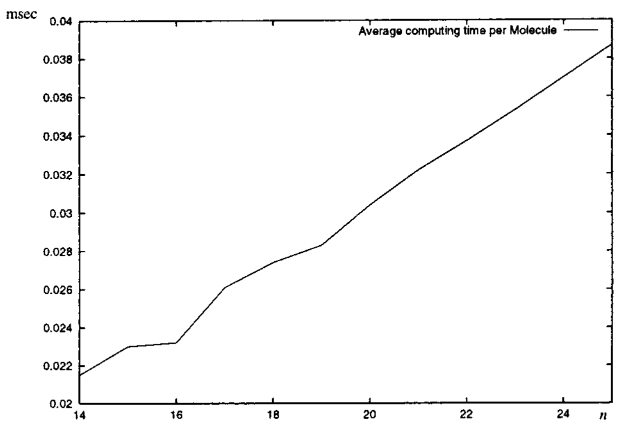

**Figure 3.** Average computing time per molecule.

**Table 5.** Statistical Indices for *WW*

| *n* | max | min | $\mu$ | $\sigma$ |
|---|---|---|---|---|
| 5 | 35 | 22 | 28.3 | 5.3 |
| 6 | 70 | 44 | 54.6 | 9.2 |
| 7 | 126 | 69 | 91.8 | 16.3 |
| 8 | 210 | 97 | 142.4 | 27.4 |
| 9 | 330 | 149 | 211.2 | 40.9 |
| 10 | 495 | 204 | 299.8 | 59.2 |
| 11 | 715 | 262 | 409.9 | 82.5 |
| 12 | 1001 | 344 | 546.0 | 111.5 |
| 13 | 1365 | 429 | 709.0 | 146.4 |
| 14 | 1820 | 517 | 903.1 | 188.9 |
| 15 | 2380 | 629 | 1130.2 | 238.6 |
| 16 | 3060 | 744 | 1393.6 | 297.3 |
| 17 | 3876 | 862 | 1695.3 | 364.9 |
| 18 | 4845 | 1049 | 2038.9 | 442.8 |
| 19 | 5985 | 1239 | 2426.5 | 531.1 |
| 20 | 7315 | 1432 | 2861.3 | 631.2 |
| 21 | 8855 | 1649 | 3345.9 | 743.2 |
| 22 | 10626 | 1869 | 3883.0 | 868.3 |
| 23 | 12650 | 2092 | 4475.6 | 1007.0 |
| 24 | 14950 | 2339 | 5126.4 | 1160.2 |
| 25 | 17550 | 2589 | 5838.1 | 1328.6 |

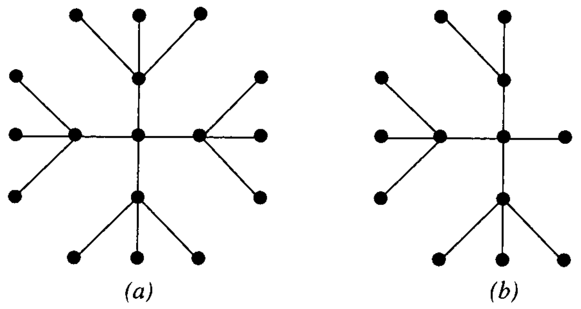

**Figure 4.** A Bethe graph (a) and an incomplete Bethe graph (b).

with $R^2 = 0.9945$. The last column reports the percentage of time spent for computing the *WW* values; the remaining time is used to read the code from the file, to translate the code into a tree and for its good numeration.

Ranges of *WW*, means, and standard deviation as functions of *n* are given in Table 5. We note that both maximum and minimum codes occurred once. As shown by Gutman et al.,[9] trees with maximum *WW* index are paths. Those with minimum *WW* index are Bethe graphs (see Figure 4a), i.e., complete ternary trees except for the fact that the root has degree 4, or incomplete Bethe graphs (see Figure 4b) in which there are some missing vertices at the last level. Note that such graphs correspond in a sense to imposition of a constraint of maximum degree equal to 4 to a star. They

**Table 6.** Distributions of *WW* Values: $n = 13, ..., 25$

| | *n* | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| intervals | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 3 | 11 | 29 | 65 | 181 |
| 5 | 0 | 1 | 2 | 3 | 8 | 29 | 60 | 154 | 406 | 1042 | 2615 | 6655 | 17412 |
| 6 | 4 | 8 | 22 | 51 | 125 | 302 | 745 | 1831 | 4594 | 11457 | 28849 | 72623 | 185221 |
| 7 | 18 | 38 | 85 | 212 | 503 | 1211 | 2980 | 7348 | 18271 | 45312 | 114054 | 288117 | 727193 |
| 8 | 37 | 86 | 200 | 480 | 1154 | 2792 | 6795 | 16686 | 40976 | 102807 | 257920 | 652629 | 1652515 |
| 9 | 58 | 139 | 328 | 771 | 1814 | 4450 | 10839 | 26927 | 67065 | 167142 | 420463 | 1061994 | 2694119 |
| 10 | 77 | 182 | 415 | 982 | 2408 | 5784 | 14350 | 35010 | 87316 | 218747 | 551021 | 1391587 | 3532177 |
| 11 | 84 | 196 | 465 | 1121 | 2651 | 6513 | 15854 | 39578 | 98824 | 246733 | 622222 | 1573337 | 3996246 |
| 12 | 90 | 200 | 472 | 1139 | 2752 | 6676 | 16347 | 40543 | 100297 | 251259 | 630523 | 1599780 | 4076522 |
| 13 | 81 | 198 | 460 | 1058 | 2599 | 6273 | 15371 | 37767 | 94150 | 237669 | 595617 | 1506480 | 3825934 |
| 14 | 73 | 164 | 384 | 960 | 2271 | 5537 | 13566 | 33863 | 84082 | 209273 | 529643 | 1336265 | 3396203 |
| 15 | 65 | 148 | 345 | 817 | 1922 | 4715 | 11707 | 28532 | 70987 | 178800 | 449143 | 1134087 | 2880795 |
| 16 | 52 | 110 | 284 | 640 | 1621 | 3897 | 9461 | 23426 | 58876 | 145895 | 366072 | 929834 | 2358674 |
| 17 | 40 | 103 | 213 | 540 | 1244 | 3147 | 7573 | 18939 | 46495 | 116107 | 293695 | 739335 | 1878659 |
| 18 | 31 | 80 | 174 | 418 | 983 | 2432 | 5928 | 14660 | 35970 | 91198 | 227942 | 576531 | 1460729 |
| 19 | 25 | 52 | 142 | 307 | 770 | 1796 | 4600 | 11058 | 27931 | 68904 | 174199 | 439285 | 1115613 |
| 20 | 21 | 42 | 105 | 243 | 590 | 1377 | 3448 | 8341 | 20890 | 52199 | 130440 | 330026 | 839644 |
| 21 | 14 | 30 | 69 | 188 | 419 | 1029 | 2418 | 6287 | 15361 | 38700 | 96839 | 244540 | 618857 |
| 22 | 9 | 28 | 51 | 132 | 317 | 764 | 1866 | 4589 | 11402 | 28047 | 70957 | 178595 | 452540 |
| 23 | 7 | 17 | 42 | 85 | 238 | 537 | 1331 | 3244 | 8074 | 20334 | 51133 | 129089 | 327263 |
| 24 | 4 | 10 | 33 | 62 | 163 | 402 | 951 | 2347 | 5845 | 14648 | 36473 | 92391 | 233831 |
| 25 | 4 | 9 | 15 | 50 | 99 | 285 | 651 | 1661 | 4098 | 10225 | 25816 | 65265 | 165423 |
| 26 | 8 | 17 | 41 | 100 | 242 | 575 | 1443 | 3257 | 8813 | 22149 | 55915 | 141915 | 361837 |

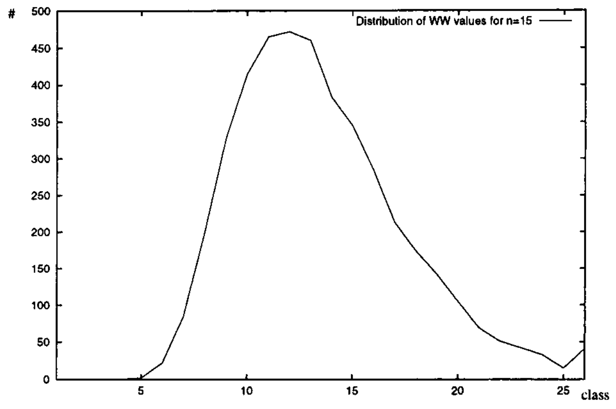

**Figure 5.** Distribution of *WW* values for $n = 15$.

appear frequently in mathematical chemistry. For instance, Bytautas and Klein[30] show that they have minimum diameter.

Further results in the distribution of *WW* values are given in Table 6 in which, for each *n* from 13 to 25, the number of values within each of 26 intervals is given; these intervals are defined as follows:

$$(\min, \mu - 3\sigma], (\mu - 3\sigma, \mu - 2.75\sigma], ..., (\mu + 3\sigma, \max)$$

a diagram of the distribution for $n = 15$ is presented in Figure 5. Distributions for other values of *n* have similar shapes: the mode is slightly to the left of the mean and there is strong right-side skewness.

## 4. CONCLUSION

A new algorithm, with a complexity linear in *n*, is proposed for computing the Hyper-Wiener index of chemical trees. This algorithm being fast, it allows finding such values for large sets of molecules, and hence computing precisely various statistics on the distribution of *WW* values.

The algorithm can be extended in several ways. First, as already noted, it applies also to general trees while still having a complexity in $O(n)$. Then, it might be used to obtain a linear algorithm for computing *WW* values for benzenoids,[31] using the isometric embedding techniques of Klavzar.[32−34] Another direction of research would be to try to extend the algorithm to tackle the case of unicyclic, or possibly bicyclic, graphs without augmenting its complexity.

## REFERENCES AND NOTES

(1) Randić, M.; Guo, X.; Oxley, T.; Krishnapriyan, H. Wiener Matrix: Source of Novel Matrix Invariants. *J. Chem. Inf. Comput. Sci.* **1993**, *33*, 709−716.
(2) Wiener, H. Correlation of Heats of Isomerization and Differences in Heats of Vaporization of Isomers among the Paraffin Hydrocarbons. *J. Am. Chem. Soc.* **1947**, *69*, 2636−2638.
(3) Wiener, H. Structural Determination of Paraffin Boiling Points. *J. Am. Chem. Soc.* **1947**, *69*, 17−20.
(4) Nikolić, S.; Trinajstić, N.; Mihalić, Z. The Wiener Index: Developments and Applications. *Croat. Chim. Acta* **1995**, *68*, 105−129.
(5) A preliminary version of this paper was presented at the Discrete Mathematical Chemistry Workshop, Rutgers University, March 23−25, 1998.
(6) Randić, M. Novel Molecular Descriptor for Structure−Property Studies. *Chem. Phys. Lett.* **1993**, *211*, 478−483.
(7) Randić, M.; Guo, X.; Oxley, T.; Krishnapriyan, H.; Naylor, L. Wiener Matrix Invariants. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 261−367.
(8) Klein, D. J.; Lukovits, I.; Gutman, I. On the definition of the Hyper-Wiener Index for Cycle-Containing Structures. *Commun. Math. Chem. (MATCH)* **1995**, *35*, 50−52.
(9) Gutman, I.; Linert, W.; Lukovits, I.; Dobrynin, A. Trees with Extremal Hyper-Wiener Index: Mathematical Basis and Chemical Applications. *J. Chem. Inf. Comput. Sci.* **1997**, *37*, 349−354.
(10) Diudea, M. V.; Parv, B. Molecular Topology. 25. Hyper-Wiener index of dendrimers. *J. Chem. Inf. Comput. Sci.* **1995**, *25*, 1015−1018.
(11) Lukovits, I. Formulas for the Hyper-Wiener Index of Trees. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 1079−1081.
(12) Lukovits, I. A note on a Formula for the Hyper-Wiener Index of Some Trees. *Comput. Chem.* **1995**, *19*, 27−31.
(13) Linert, W.; Renz, F.; Kleestorfer, K.; Lukovits, I. An Algorithm for the Computation of the Hyper-Wiener Index for the Characterization and Discrimination of Branched Acyclic Molecules. *Comput. Chem.* **1995**, *19*, 395−401.

(14) Lukovits, I.; Linert, W. A Novel Definition of the Hyper-Wiener Index for Cycles. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 899−902.

(15) Diudea, M. V. Wiener and Hyper-Wiener Numbers in a Single Matrix. *J. Chem. Inf. Comput. Sci.* **1996**, *36*, 833−836.

(16) Diudea, M. V. Walk Numbers $^{e}W_{M}$: Wiener-Type Numbers of Higher Rank. *J. Chem. Inf. Comput. Sci.* **1996**, *36*, 535−540.

(17) Diudea, M. V. Indices of Reciprocal Properties of Harary Indices. *J. Chem. Inf. Comput. Sci.* **1997**, *37*, 292−299.

(18) Diudea, M. V.; Katona, G.; Minailiuc, O. M.; Parv, B. Molecular Topology. 24. Wiener and Hyper-Wiener Indices in Spiro-Graphs. *Russ. Chem. Bull.* **1995**, *44*, 1606−1611.

(19) Linert, W.; Kleestorfer, K.; Renz, F.; Lukovits, I. Description of Cyclic and Branched-Acyclic Hydrocarbons by variant of the Hyper-Wiener Index. *J. Mol. Struct. (THEOCHEM)* **1995**, *337*, 121−127.

(20) Linert, W.; Lukovits, I. Formulas for the Hyper-Wiener and Hyper-Detour Indices of Fused Bicyclic Structures. *Commun. Math. Chem. (MATCH)* **1997**, *35*.

(21) Zhu, H. Y.; Klein, D. J.; Lukovits, I. Extensions of the Wiener Number. *J. Chem. Inf. Comput. Sci.* **1996**, *36*, 420−428.

(22) Berge, C. *Graphes et Hypergraphes*; Dunod: Paris, 1970.

(23) Ahuja, R. K.; Magnanti, T. L.; Orlin, J. B. *Network Flows. Theory, Algorithms, and Applications*; Prentice Hall: Englewood Cliffs, NJ, 1993.

(24) Aho, A. V.; Hopcroft, J. E.; Ullman, J. D. *Data Structures and Algorithms*; Prentice Hall: Addison-Wesley: 1983.

(25) Available on line via anonymous ftp at ftp.di.unipi.it/pub/project/orgroup/aringhie/freealkanes.zip.

(26) Read, R. C. The Coding of Various Kind of Unlabeled Trees. In *Graph Theory and Computing*; Academic Press: New York, 1972; pp 153−182.

(27) Hansen, P.; Jaumard, B.; Lebatteux, C.; Zheng, M. Coding Chemical Trees with the Centered *N*-Tuple Code. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 782−790.

(28) Aringhieri, R. Metodi dell'Ottimizzazione Combinatoria applicati alla Chimica: L'Eumerazione ed il Calcolo del Numero di Hyper-Wiener di Molecole di Alcano. *Tesi di Laurea*; Dipartimento di Informatica, Università di Pisa, 1996.

(29) Kvasnicka, V.; Pospichal, J., Constructive Enumeration of Acyclic Molecules. *Collect. Czech. Chem. Commun.* **1991**, *56*, 1777−1802.

(30) Bytautas, L.; Klein, D. J. Alkane Isomer Combinatorics: Stereostructure Enumeration and Graph-Invariant and Molecular-Property distributions. *J. Chem. Inf. Comput. Sci.* **1999**, *39*, 803−818.

(31) Klavzar, S. Private Communication at the Discrete Mathematical Chemistry Workshop; Rutgers University, March 23−25, 1998.

(32) Chepoi, V.; Klavzar, S. The Wiener Index and the Szeged Index of Benzenoid System in Linear Time. *J. Chem. Inf. Comput. Sci.* **1997**, *37*, 752−755.

(33) Klavzar, S. Application of Isometric Embeddings to Chemical Graphs. In *Discrete Mathematical Chemistry*; Hansen, P., Fowler, P., Zheng, M., Eds.; *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*; 2000; Vol. 51, pp 249−259.

(34) Klavzar, S.; Zigert, P.; Gutman, I. An Algorithm for the Calculation of the Hyper-Wiener Index of Benzenoid Hydrocarbons. *Comput. Chem.* **2000**, *24*, 229−233.

CI0001536