Università di Pisa

Dipartimento di Informatica

Technical Report : TR-99-08

A Linear Algorithm for the Hyper-Wiener Number of Chemical Trees

R. Aringhieri, P. Hansen, F. Malucelli

April 14, 1999

ADDR: Corso Italia 40,56125 Pisa,Italy. TEL: +39-50-887111. FAX: +39-50-887226

A Linear Algorithm for the Hyper-Wiener Number of Chemical Trees.

R. Aringhieri[1], *P. Hansen*[2] *and F. Malucelli*[3]

Abstract: *The paper addresses the problem of computing the Hyper-Wiener index for molecules in the alkanes family. A new algorithm having linear computational complexity is proposed. The algorithm has optimal complexity, and can be extended to the family of all the acyclic molecular structures. Some computational results are reported.*

1. Introduction

In computational chemistry, the determination of indices which synthesize the information about structural properties of the molecules has received a great deal of importance. Molecules with the same atoms, but with different structures may have very different physical and chemical properties, as for example the fusion and boiling points, conductability, osmotic pressure, crioscopic characteristics, and so on. One measure that can capture these structural differences is the Wiener index (W) [8, 9], and its generalization called Hyper-Wiener index (HW) [4, 5]. These two numbers measure the *compactness* of acyclic structured molecules.

In the present paper we consider the computation of HW and we propose an algorithm to compute it efficiently. In particular we focus on the case of the alkanes molecular family, however the algorithm can be easily generalized to any other acyclic molecular structure.

The alkane molecular family is composed by *classes of homologous molecules*, that is molecules with the same numbers of carbonium and hydrogen atoms; the n-th class is characterized by the following formula:

$$C_n H_{2n+2}, \qquad n=1, 2, \ldots$$

An alkane molecule is usually represented by indicating the carbonium atoms and their (primary) links, omitting to represent hydrogen atoms (see figure 1). When n is sufficiently large ($n \geq 4$), to the same formula correspond molecules with different structures. Take as an example *butane* ($C_4 H_{10}$): there are two possible molecular structures *nbutane* and *isobutane* as represented in figure 1.a and 1.b, respectively.

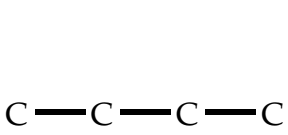


figure 1.a: nbutane

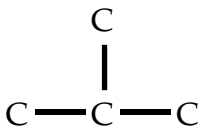


figure 1.b: isobutane

Molecules of the same class but with different structure have different compactness measures (HW) and different chemical and physical properties. In the above case, we

---

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

2 GERAD - HEC, Université de Montréal, 5255 Avenue Decelles, Montréal, H3T 1V6 (Canada)

3 ISTEL - Università di Perugia, S.Lucia Canetola - 06131 Perugia - Italy

have that HW of nbutane is 15 and HW of isobutane is 12, and for example their boiling points are 0°C and -12°C, respectively.

Alkanes can be represented by acyclic connected graphs whose nodes correspond to carbonium atoms and edges correspond to the primary links. Thus the family of alkanes correspond to the family of the connected acyclic graphs (i.e. *trees*) whose nodes have degree less than or equal to 4.

The problem of efficiently enumerating all the alkanes of a class has been studied since nineteenth century. To enumerate degree constrained trees, the graph must be suitably encoded. Different numerical codes have been proposed as well as a good number of efficient algorithms to produce all the codes of one alkane class [1, 2, 6, 7].

The paper is organized as follows: in Section 2 we formally define the Wiener and Hyper-Wiener numbers W and HW, and we present some properties. In Section 3 we discuss an efficient algorithm to compute all the HW numbers of a class of alkanes. The proposed algorithm improves on the computational complexity of the known algorithms of the literature, and we prove that it has optimal complexity. Section 4 reports some computational results.

2. The Hyper-Wiener number

A topological index to measure the compactness of a molecule was first introduced by Wiener [9]. Let us consider the graph representation of a molecule described in the previous Section. That is, a molecule can be represented by a tree $T=(V,E)$ where the nodes in V correspond to carbonium atoms, and the edges in E to the primary links. For each edge $(i,j)\in E$, consider the two connected components $T_i=(V_i,E_i)$ and $T_j=(V_j,E_j)$ obtained by removing edge (i,j) from T. The Wiener number W is given by:

$$W = \sum_{i,j\in E} |V_i||V_j|. \tag{2.1}$$

For the nbutane and the isobutane we obtain W=10 and W=9, respectively.

The study of significant topological indices has been deepened by Randic, Guo, Oxley and Krishnapriyan [5]. They extended and generalized the Wiener number by introducing the Wiener matrix. The extension considers all the pairs of nodes (i,j) instead of considering only the adjacent ones. In particular, the generic entry w_{ij} of the Wiener matrix is defined by:

$$w_{ij} = |V_i^{ij}||V_j^{ij}|, \qquad \forall\ i, j \in N,\ i\neq j, \tag{2.2}$$

where V_i^{ij} and V_j^{ij} are the set of nodes in the connected components containing i and j obtained by removing from T the unique chain P_{ij} between i and j. Note that the Wiener matrix is symmetric.

The Hyper-Wiener number HW is defined as follows [5]:

$$HW = \sum_{i\in N} \sum_{j\in N,\, i<j} w_{ij} = \sum_{i\in N} \sum_{j\in N,\, i<j} |V_i^{ij}||V_j^{ij}| \tag{2.3}$$

The computation of HW seems to require $O(n^2)$ time, where n is the number of nodes of the tree, as it is the sum of $n(n\text{-}1)/2$ elements. In the next Section we show that this computation can be carried out in $O(n)$ time.

The following theorem gives an upper bound to HW for graphs with n nodes. A chain $C=(V,E')$ is a particular tree whose nodes have degree less than or equal to 2.

Theorem 2.1 [3]
Consider a tree $T=(V,E)$ with $|V|=n$. The Hyper Wiener number of T is less than or equal to Hyper Wiener number of the chain $C=(V,E')$. Moreover the Hyper Wiener number of a chain C with n nodes is

$$HW = n(n+1)(n(n+1)\text{-}2)/24. \qquad (2.4)$$

❑

In practice the above theorem confirms the intuitive idea that the less compact molecule is the one with a chain structure.

3. The algorithm for HW computation

In order to devise an efficient algorithm for the computation of HW, we have to rewrite (2.3). First, note that we are dealing with trees with a particular structure. This implies that we can renumber the nodes of the tree and orient the edges in such a way that there is always an oriented path from any node i to node $n = |V|$ and

$$\text{if } (i,j) \in V \text{ then } i < j,$$

where V is the set of oriented arcs. This is called *good numeration* of the graph. It is always possible to find a good numeration of a tree; a possible way is described by the following algorithm.

Algorithm Good Numeration
Input: $T=(V,E)$; Output: Label;
0. S be an empty stack; **for each** $u \in V$ **do** Visited(u):=false;
   Let u such that Degree(u)=1;
   Label(u):=n; Next_Number:=n-1; Visited(u):=true; **for each** $v \in$ Adjacent(u) **do** Push(v,S);
1. u:=Pop(S); Label(u):=Next_Number; Next_Number:=Next_Number-1; Visited(u):=true;
   for each $v \in$ Adjacent(u) such that not Visited(v) **do** Push(v,S);
2. **if** S is empty **then** stop **else** go to 1.

We assume that the nodes adjacent to any node u (i.e. those contained in Adjacent(u)) are sorted by non decreasing degree. The output vector Label contains the new numeration of the nodes. From now on, let us assume that the graph is well numbered.

Now, observe that $|V_i^{ij}|$ does not depend on j, if $j > i$; in fact in the connected component V_i^{ij} there are only nodes with number less than or equal to i, and any path going from i to any j ($>i$) does not involve any one of these nodes (according to the good numeration).

Thus (2.3) can be rewritten as follows:

$$HW = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} |V_i^{ij}||V_j^{ij}| = \sum_{i=1}^{n-1} |V_i^{ij}| \sum_{j=i+1}^{n} |V_j^{ij}|. \tag{3.1}$$

If we define $M_i = \sum_{j=i+1}^{n} |V_j^{ij}|$, the HW number can be computed as follows:

$$HW = \sum_{i=1}^{n-1} |V_i^{ij}| M_i \,. \tag{3.2}$$

The linearity of the computation of HW depends on the fact that, for a given i, M_i can be obtained recursively in constant time knowing the values M_j, $j>i$. In order to do this computation we have to distinguish several cases. Let us begin with the simplest case, the chain case. Consider the computation of w_{ij} and assume that between i and j there are no other nodes than those in the chain P_{ij}, that is, by removing the edges of P_{ij} we obtain only two connected components. If we exclude the case of P_{ij} equal to a single edge (i.e. (i,j)), we have the situation depicted in fig. 2, where h can be possibly equal to k.

figure 2

In this case, obviously V_i^{ij} is equal to V_i computed with respect to edge (i,h), and V_j^{ij} is equal to V_j computed with respect to edge (k,j). This suggests us to compute all the values of $|V_i|$ and $|V_h|$ for each edge (i,h) in order deal with these cases very efficiently. The algorithm considers in input the tree T, and gives in output the labels $C(i,j).i$ and $C(i,j).j$ on each edge (i,j), whose values are $|V_i|$ and $|V_j|$, respectively, computed with respect to the elimination of edge (i,j).

```
Algorithm Edge Labels Computation:
  Input: Tree T=(V,E);  Output: C(i,j).i and C(i,j).j for each (i,j)∈E.
  begin
    for each i∈V do t(i):=1;
    repeat
      for each (i,j)∈E such that Degree(i)=1 do
      begin
        C(i,j).i := t(i); C(i,j).j :=n - t(i);
        E:=E\{(i,j)};
        t(j) := t(j) + C(i,j).i
      end
    until E = Ø;
  end
```

The computational complexity of this algorithm is $O(n)$ as the edges are considered only once.

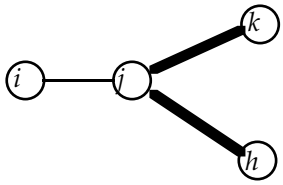
So, for the chain case, we have that:

$$M_i = M_j + C(i,j).j.$$

$$M_{n-1}=1 \tag{3.3}$$

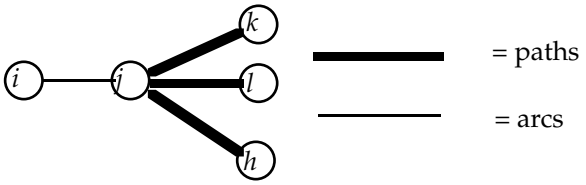
Note that the above formula establishes a visit order of the tree necessary to compute the values of M_i. In fact node j must be visited after node i.

In the case where node j has degree 3 or 4 the computation of M_i needs to consider all the nodes adjacent to j. The situation is depicted in figures 3 and 4.



$i+1=k<j<h$

figure 3



$i+1=k<l<j<h$

figure 4

In order to maintain the information about all the paths incident with node j we introduce the value M_j^{aux}.

$$M_j^{aux} = \sum_{(v,w)\in H} C(v,w).v \tag{3.4}$$

where $H =\{(v,w)\in P_{kj}$: degree$(k)=1, k<j\}$.

Also M_j^{aux} can be calculated incrementally as in the case of M_j. In the case of figure 3 (node j of degree 3) we have:

$$\begin{aligned} &M_w^{aux} = M_v^{aux} + C(v,w).v \qquad w=k+1,\ldots,j, \\ &M_k^{aux} = 0. \end{aligned} \tag{3.5}$$

In the case of figure 4 (node of degree 4 or more) we have:

$$\begin{aligned} &M_w^{aux} = M_v^{aux} + M_w^{aux} + C(v,w).v \qquad w=k+1,\ldots,j, \\ &M_k^{aux} = 0. \end{aligned} \tag{3.6}$$

Therefore, we have that:

$$M_i = M_j + C(i,j).j + M_j^{aux}. \tag{3.7}$$

Hence the computation of M_i is based not only on the path coming from the root of the tree visit but also on the backtracking paths incident with j. Obviously those paths can be easily managed with a Depth First Search on the tree. The algorithm that computes the Hyper Wiener number is a variant of a DFS visit of the tree and is summarized in the following:

```
Algorithm Hyper Wiener:
  Input: Tree T=(V,E);  Output: HW
  begin
    for each i∈ V do
    begin
      M(i):=0; Maux(i):=0; visited:=false;
    end;
    curr_node:=n-1;
    prec_node:= n;
    visited[n]:= true;
    visited[n-1]:=true;
    M[curr_node]:=1;
    while (curr_node≠1) do
    begin
      if ∃u∈ Adjacent[curr_node] such that visited[u]=false and u<curr_node then
      begin
        prec_node:=curr_node;
        curr_node:=u;
        M[curr_node]:=M[prec_node]+C(curr_node,prec_node).prec_node+Maux[prec_node];
      end
      else begin
        temp:=curr_node;
        curr_node:=prec_node;
        prec_node:=temp;
      Maux[curr_node]:=Maux[curr_node]+Maux[prec_node]+C(prec_node,curr_node).prec_node
      end
    end;
    fori:=1,…,n-1 do
      HW:=HW+C(i,i+1).i  M[i]
  end.
```

Note that the algorithm is a particular DFS visit of T where some arcs are visited twice. Given that we visit each arc at most twice and each vertices at most three times, the algorithm Hyper Wiener computes the values M_i in O(n) time, hence it gives HW in O(n). Observe also that a lower bound to the time complexity is O(n) as the size of input tree is O(n). Therefore, we conclude that the proposed algorithm has optimal complexity. Finally, we present an example of the complete algorithm.

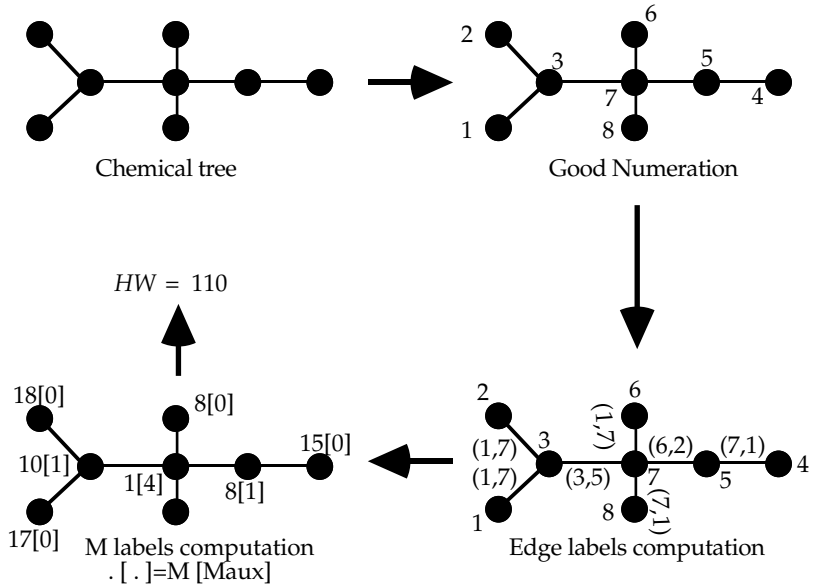


figure 5

4. Computational results

In this section we report some computational times and statistical analysis results on the distribution of *HW*. The results are obtained on a SunOS 4.1.3 U1 4m by applying the algorithm to an alkane molecular family contained in a file of numerical codes.

n	# of codes	Total time	Avg. time per molecule	% time for HW comp.
14	1858	1083	0.5828	59.8
15	4347	2800	0.6441	58.8
16	10359	7070	0.6824	60.5
17	24894	19640	0.7886	58.4
18	60523	47730	0.7889	59.4
19	148284	120920	0.8154	58.9
20	366319	301450	0.8230	60.1

Table I: Computing times (in msec.)

It should be noted that the average time increases with the size of the molecule rather than being constant: in fact the number of molecules does not grow proportionally with respect to *n*. The last column reports the percentage of the total time spent computing the *HW*. The remaining time is used to translate the numerical code in graphs and for its numeration.

A statistical study of the alkane molecular family is important to derive some common properties of *HW*. In Table II we show the result of the standard statistical indices: μ, σ , maximum and minimum.

n	max	min	μ	σ
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

Table II: Statistical indices

Note that the minimum *HW* correspond to the value stated by (2.4).

n	10%	20%	30%	40%	50%	60%	70%	80%	90%
10	232	248	263	275	290	304	324	343	383
11	317	337	361	376	398	419	441	474	520
12	414	450	476	501	530	557	589	629	696

Table III: Deciles for HW'distribution

Figures 5, 6, and 7 report the distribution of the *HW* in the classes of n=10, 11, 12 based on the deciles reported on the table III.

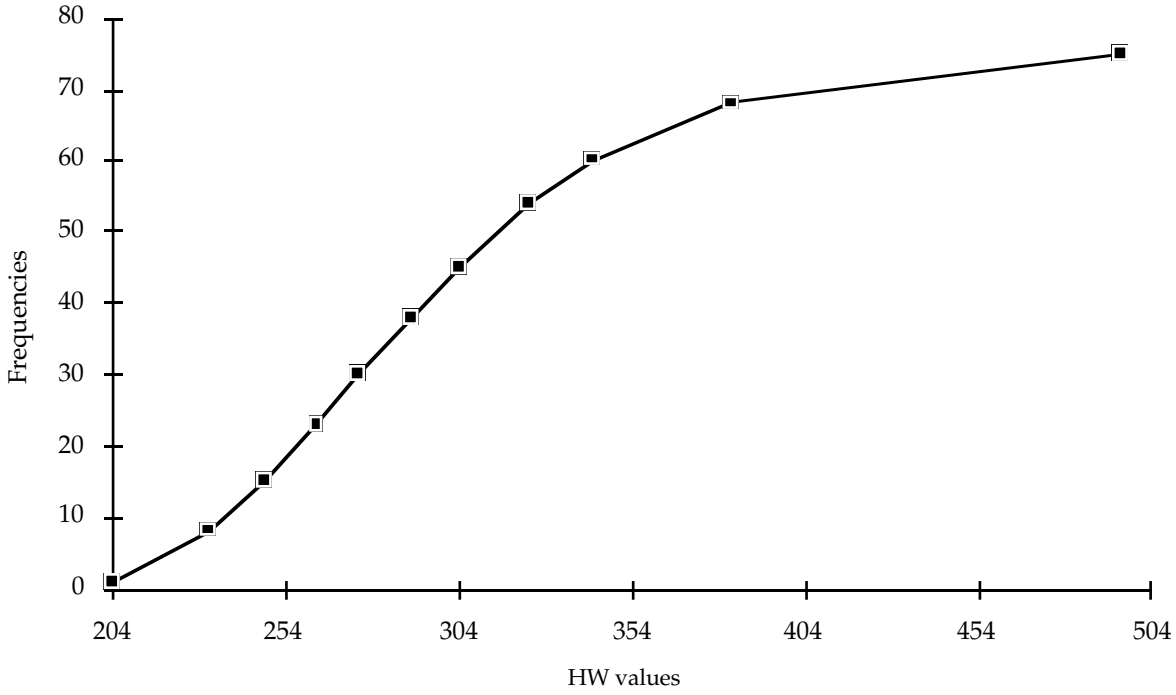


Figure 6: Distribution for n=10

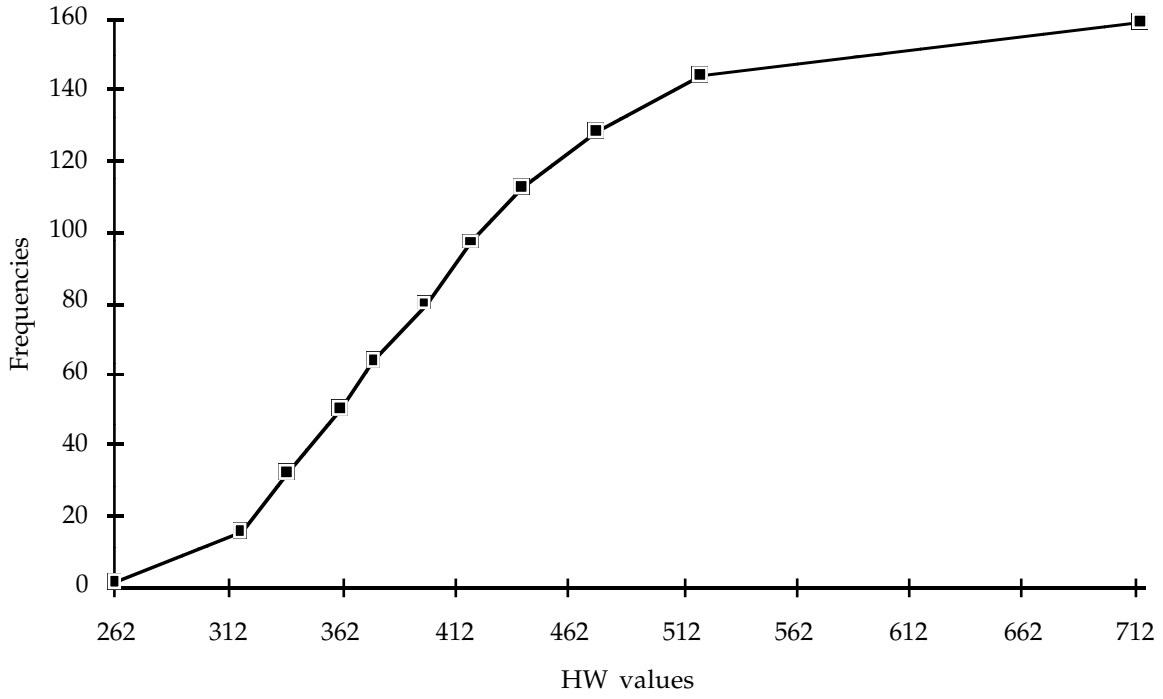


Figure 7: Distribution for n=11

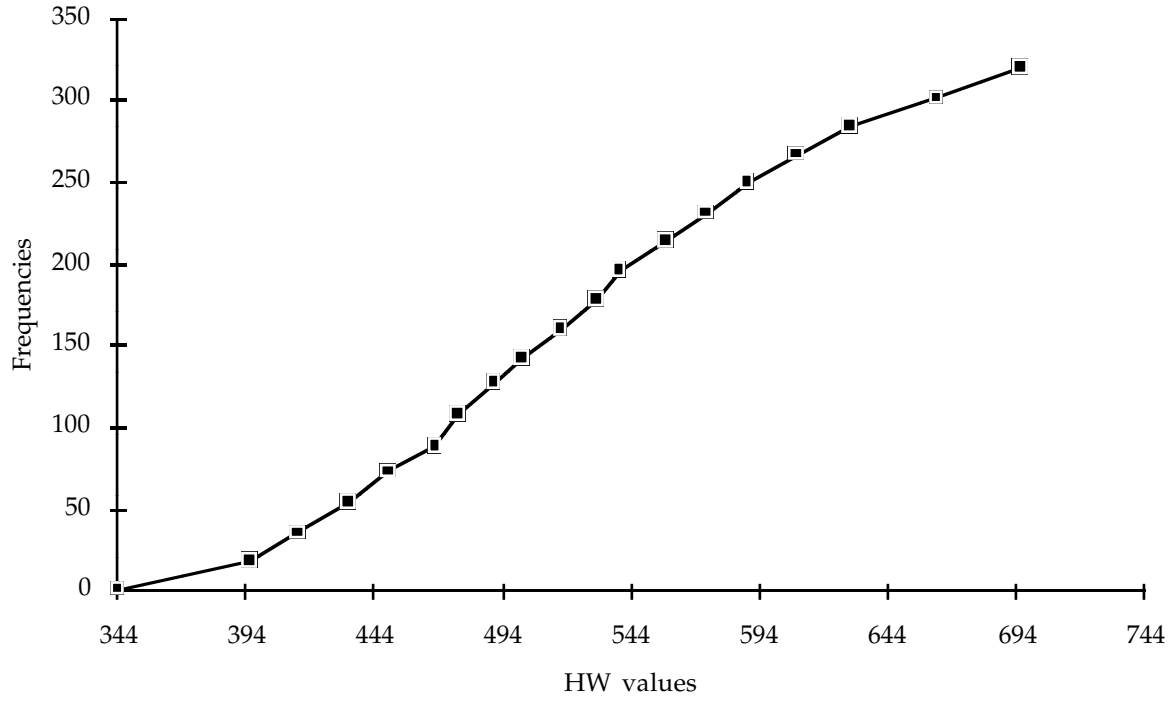


Figure 8: Distribution for n=12

References

[1] Aringhieri, R., *Metodi dell'Ottimizzazione Combinatoria Applicati alla Chimica: L'Enumerazione ed il Calcolo del Numero di Hyper-Wiener di Molecole di Alcano*, (1996) Dipartimento di Informatica - Università di Pisa.

[2] Hansen, P., B. Jaumard, C. Labatteux and M. Zeng, *Coding Chemical Trees with the Centeres N-Tuple Code.* **J. Chem. Inf. Comput. Sci.**, 1994. 34: p. 782-790.

[3] Lukovits, I., *Formulas for the Hyper-Wiener Index of Trees.* **J. Chem. Inf. Comput. Sci.**, 1994. 34: p. 1079-1081.

[4] Randic, M., *Novel Molecular Descriptor for Structure-Property Studies.* **Chem. Phys. Lett.**, 1993. 211: p. 478-483.

[5] Randic, M., X. Guo, T. Oxley and H. Krishnapriyan, *Wiener Matrix: Source of Novel Matrix Invariants.* **J, Chem. Inf. Comput. Sci.**, 1993. 33: p. 709-716.

[6] Read, R. C., *The Coding of Various Kind of Unlabelled Trees in Graph Theory and Computing*, . 1972, Academic Press: New York. p. 153-182.

[7] Trinajstic, N., S. Nicolic, J. V. Knop, W. R. Muller and K. Szymanski, *Computational Chemical Graph Theory*. 1991, New York: Ellis Horwood.

[8] 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: p. 2636-2638.

[9] Wiener, H., *Structural Determination of Paraffin Boiling Points.* **J. Am. Chem. Soc.**, 1947. 69: p. 17-20.