Università di Pisa

Dipartimento di Informatica

Technical Report : TR-99-09

Chemical Tree Enumeration Algorithms

R. Aringhieri, P. Hansen, F. Malucelli

April 14, 1999

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

Chemical Tree Enumeration Algorithms.

Roberto Aringhieri *

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

Pierre Hansen

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

Federico Malucelli

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

April 14, 1999

Abstract

The need to generate graphs representing chemical structures suggested the use of computer and as a consequence the delevopment of efficient enumerating algorithms. This paper considers the isomeric acyclic structures enumeration focusin on the alkane molecular family. Two new algorithms based on **Reverse Search** [7] [8] are presented. The main concern of this paper is to highlight the importance of the asymptotic complexity and running time behaviour in the analysis of enumeration algorithms.

1 Introduction.

The enumeration of the alkane molecular family is a problem which has root in the nineteenth century. Alkanes are chemical compounds with identical molecular formulae and weights but differing at least in some physical and/or

*Corresponding author. Email: raringhieri@di.unipi.it

chemical properties leading to different structures. Alkanes belong in the general class of isomeric acyclic structures. The term isomerism and the definition of isomers were introduced in 1830 by Berzelius [11].

The alkane molecular family partitioned into *classes of homologous molecules*, that is molecules with the same numbers of carbonium and hydrogen atoms; the nth class is characterized by the following formula:

$$C_nH_{2n+2}, \quad n = 1, 2, \ldots \tag{1}$$

An alkane molecule is usually represented by indicating the carbonium atoms and their (primary) links, omitting to represent hydrogen atoms (see figure 1, 2 and 3). When n is sufficiently large ($n \geq 4$), to the same formula correspond molecules with different structures. Take as an example butane (C_4H_{10}): there are two possible molecular structures nbutane and isobutane as represented in figure 2 and 3, respectively.

Alkanes can be represented by acyclic connected graphs whose nodes correspond to carbonium atoms and edges correspond to the primary links. Thus there is a one-one correspondence between elements in the family of alkanes and elements in the family of the connected acyclic graphs (i.e. trees) whose nodes have degree less than or equal to 4. As a consequence, an alkane molecula can be represented by a degree constrained tree.

Cayley was the first to realize the potential of the mathematical theory of trees for the enumeration of isomeric acyclic structures [12] [13] [14]. He has enumerated the alkane isomers and alkyl radicals with up to $n = 13$ with some counting errors. His work had a considerable impact on chemists of his time. Then the development took a turn to computer-oriented methods for the enumeration and generation of isomeric structures: the development of techniques for the generation of graphs (chemical structures) by computing devices has made possible the direct enumeration of isomeric acyclic structures by way of enumerations algorithms. The pioneering contributions in this area are due to Lederberg and his group [26] [27] and Corey and his group [16] [17].

In this paper we introduce two new algorithms for the generation of the alkanes molecular family based on **Reverse Search** [7] [8]. We study algorithms correctness and their computational performance, moreover we compare them with other enumeration algorithms. Our main concern is to highlight the importance of the asymptotic complexity and running time behaviour in the analysis of enumeration algorithms.

In order to efficiently enumerate degree constrained trees, data must be suitably encoded: the motivation is that a code can be easlyr handled by computer program. Read [29] lists some desirable properties for a molecular code such as being linear, unique, well-defined, brief, simple, easily comprehensible to chemists, and efficiently encodable and decodable. Different numerical codes have been proposed in literacture [9] [18] [23] [29] [30] [31]. In the following, we usually refer to N-tuple code. For more details about its computation, the reader can refer to [29] [31].

The paper is organized as follows: in section 2 we describe two alkane enumeration algorithms founded in literature; in section 3, we present two new algorithms which differ from the point of view of the computer implementation; the asymptotic complexities of the presented algorithms is analyzed in section 4; the running time behaviour are compared in sections 5. Finally, we devise some conclusions and draw some directions for future work in section 6. Hereafter, for the sake of simplicity, we use the expression "tree" or "alkane" rather than "degree constrained tree". We use the notation n-tree, n-rooted to denote respectively a tree and a rooted tree with n nodes.

2 Enumeration algorithms

In this section we shortly describe two algorithms [31] [24] based on two different ways to generate alkanes.

The algorithm described in [31] is based on the following idea: a n-tree can be obtained getting a $(n-1)$-tree and adding one node in such way that the degree constraint still holds. In the following we refer to this class of algorithms as *one-to-one enumeration*.

The algorithm does not guarantee to avoid duplications, as shown in figure 4 where is given an example of tree enumeration passing from methane to penthane: in the figure, each edge represents from how a tree can be generated and each label on it gives the number of different ways in which it can be obtained. The enumeration of a tree twice or more times from the same tree or from more than one tree is the main problem emerging from this method. Therefore the one-to-one approach requires the introduction of some rules which guarantee the uniqueness of the generated trees. In [31] these rules are not specified. In the following, we refer to this algorithm as TNKMS.

The *constructive enumeration* algorithm is based on the following remark:

a n-tree can be decomposed in one node and one or more rooted trees; in the same way each rooted tree can be decomposed and so on. Going in the reverse way, a n-tree can be built joining together one node and one or more rooted trees such that the sum of nodes is equal to n. An example is given in figure 5 in which the tree a and b are built respectively joining to the squared node two 2-rooted and one 1-rooted, three 1-rooted and one 2-rooted. In [24] these ideas are described from a more formal and mathematical point of view.

The rules stated in Jordan's theorem guarantee the unique enumeration, that is the avoidance of duplications.

Theorem 1 (Jordan 1869) *Let T be a n-tree, the following three different cases should be separately considered:*

1. *For odd $n = 2k+1$ there exist a unique node, called the* **centroid***, such that all (two or more) incident subtrees are composed, at most, of k nodes.*

   For even $n = 2k$ there exist either

2. *a unique node - centroid such that all (three or more) incident subtrees are composed of less than k nodes, or*

3. *a unique edge, called the* **bicentroid***, such that the incident two subtrees are composed exactly of k nodes.*

For example, to build the set of 10-trees, we have to compose together one node and three or more rooted trees with less than 4 nodes in such a way that the sum of nodes are equal to 9. Building degree constrained trees, we have to take into account that we are not allowed to compose more than four subtrees. Moreover, we join together two rooted trees with 5 nodes. In figure 5, c, d and e are the bicentroid trees with $n = 6$. These compositions are repeated for all possible combinations of trees.

The choice of the trees depends only on their cardinality. So, finding t trees with no more than k is equivalent to finding t numbers such that their sum gives $n-1$ and each of them is less than or equal to k.

k has to be greater than or equal to 2 according to the Jordan's theorem and less than or equal to 4 according to the degree constraint.

We could not find any computational result about the implementation of this approach, neither we coud find any program implementing it, therefore we made our own implementation following the indications given by the authors in the original paper. In the follows, we refer to this algorithm as LKP.

3 Two new enumeration algorithms

In this section we present two new algorithms [6] [19] belonging in the class of one-to-one enumeration which are characterized by the different molecular codes used in the computer implementation [2].

As discussed above, one-to-one algorithms require some rules which guarantee the uniqueness of the enumeration. A first attemps can be carried out by implementing a list of enumerated trees: a new tree is accepted if and only if it does not exist in the list. The use of a list, ordered or not, requires a large amount of computational resources, in terms of time and memory space.

In order to avoid the use of a list, we introduced two techniques which detect duplicated trees, namely *Reverse Search* and *Symmetry techniques*.

First of all, we introduce the trees generating function G. Let T_o, T_d, O, D be respectively an origin tree, a derived tree, the set of origin trees and the set of derived trees. The tree generating function is a function $G : O \rightarrow D$ such that for each $T_o \in O$ returns the set of T_d which can be derived from it according to degree constraint. G^{-1} is the inverse operator: it returns the set of T_o generating $T_d \in D$.

The Reverse Search was introduced by Avis and Fukuda in their work on polyhedra vertex enumeration [7] [8]. It consists of an acceptance rule and involves T_o and T_d: $T_d \in G(T_o)$ is accepted if and only if $T_o = \max(G^{-1}(T_d))$. In other words, $T_d \in G(T_o)$ is accepted if and only if T_o is the greatest in $G^{-1}(T_d)$ with respect to some ordering on the origin tree set.

The Reverse Search generates a unique n-tree if and only if each tree in the $(n-1)$-tree origin set are unique. So, it is necessary to start the enumeration from an origin set such that this uniqueness property holds, for example from C_3H_8.

The Symmetry technique is the way to recognize identical subtrees in the origin tree. It is obvious that identical subtrees in T_o give raise to the same derived tree T_d. The symmetry idea is shown in figure 6 in which the symmetries arising in the 4-trees are represented.

These two techniques guarantee the uniqueness in the enumeration process as depicted in figure 7, where they are applied to the last level of figure 4.

From a we derive twice both c and d and from b we derive once e and three times d; checking symmetries (depicted in figure 6) in a and b we derive once c and d and applying the acceptance rule we derive once d from a or b.

Centered N-tuple code has been introduced by Hansen *et al.* [18] as a new way of coding chemical tree. The Hansen's code is similar to the classic N-tuple one. Indeed both codes select one node as the root of the coding computation. In N-tuple coding, the N-tuple code is computed for each node with maximum degree and then the maximum ones are considered. In Centered N-tuple coding, the N-tuple code is computed on center nodes [1] and then the maximum one is considered.

In the worst case, Centered N-tuple requires the computation of two codes while N-tuple computes $n-2$ codes, in the case of the chain tree. Thus the Centered N-tuple code is more efficient to compute than N-tuple code. Actually, the authors in [18] show that Centered N-tuple code can be computed in $O(n \log n)$ while N-tuple code in $O(n^2)$.

The algorithm developed in [6] uses the centered N-tuple code while [19] uses the N-tuple code.

The centered N-tuple code allows to implement both the Reverse Search and the Symmetry techniques quite easily. We refer to it as AHM.

The Reverse Search acceptance rule becomes the following: $T_d \in G(T_o)$ is accepted if and only if T_o is the maximum in $G^{-1}(T_d)$ with respect to lexicographic order.

The investigation of symmetries in trees requires a data structure called CTree, the code tree: CTree is a rooted tree whose nodes correspond to the subtree code. An example is given in figure 8. We observe that at same level the subtree codes are ordered from the maximum (on the left) to the minimum (on the right) with respect to the lexicografic order.

The investigation of symmetries in CTree is quite easy. Performing a breadth-first search on CTree, a symmetry is equivalent to find two nodes in CTree with same code at the same level. When a symmetry is detected, a breadth-first search is performed only in one of the two subtrees. In figure 8, we have one symmetry between 10 and 10 at level 2.

The CTree do not allow to check the symmetry between the two subtrees as in the bicentroid case (see Jordan's theorem). The simpliest way to recognize it is during the generation: if we are generating a tree in such way, we put to 1 a flag associated to the tree.

The generation of a new tree requires also its encoding. The new code can be computed in $O(n \log n)$ using the procedure reported in [18]. We are going to show that this computation can be carried out in linear time. First of all, we can state the following rule:

> **[Recoding Rule]**: Let $c(v) = s_v \oplus c_1 \oplus \ldots \oplus c_{s_v}$ the centered N-tuple code of the rooted tree in v in which s_v is the first digit of c_v and c_i is the code of ith subtrees, $i = 1, \ldots, s_v$. We suppose to add one node as leaf of v. Then, the new $c(v)$ is equal to $(s_v + 1) \oplus c_1 \oplus \ldots \oplus c_{s_v} \oplus 0$.

Adding a new leaf at level j in CTree modifies the subtree code which can be computed by Recoding Rule. The modified code can imply a different lexicographic order among the codes at level j belonging to the same code at level $j-1$, that is the codes having the same subtree root. This means that the modified code has to be compared with the codes on its left having the same root.

Finally, also the addition affects the lexicographic order among the subtree codes at level $i = 0, \ldots, j-1$. So, the code of T_d can be computed using the Recoding Rule at level j and then checking the lexicographic order between the modified subtree code and the subtree codes on its left having the same root, and repeating the check at each level until the root is reached. The procedure is described in figure 9.

The same idea of procedure can be used to compute the code of all possible origin trees since an origin code tree can be obtained from CTree by eliminating its leafs according to the symmetry idea also applied in this case: the leafs belonging to the two or more subtrees having the same code, give raise to the same origin tree. An example is given by figure 9, reading it from the left to the right in which 22101021010 has only one origin tree.

The tree center can change by adding or eliminating one leaf. In [6] some rules which detect this changement are described. They are based on the following general remark: at least two subtrees for which the lenght between their deepest leaf and the node center(s) is equal to the maximum depth (maximum minus 1), must leave from the tree center(s). Hence the center can change only if we modify this situation.

In section 4, we formally show that these procedures have linear time complexity.

The algorithm presented in [19] uses the N-tuple code to implement the Reverse Search and the Symmetry techniques. The acceptance rule does not change using the N-tuple code. To recognize simmetries and to compute the new code, the authors implement the above ideas directly on the origin code and they do not use something like the CTree data structure. In the following, we refer to this algorithm as HPVK. As we show in the next section, the two algorithms have the same asymptotic complexity.

Hereafter we report a pascal-like procedure **Gen** which generates the set of derived n-trees given the origin $(n-1)$-tree. Let *GS*, *InvGS*, *get* be the generating function which recognize symmetries, the *GS* inverse function, the function which extracts one element from a set.

```
procedure Gen (T,n);
begin
   D,O:sets of trees;
   D:=GS(T);
   while D is not empty
   do begin
      t:=get(D); O:=InvGS(t);
      if code(t) = max(code(O)) then
         begin put t in C(n);
               n_of_trees[n]:=n_of_trees[n]+1
         end else delete t;
   end;
end.
```

4 Asymptotic complexities

In this section, we provide an asymptotic complexity analisys for AHM, HPVK and LKP algorithms.

As for the AHM, first we provide a proof of the computation of a new tree code is linear and then we analyze the complexity of whole procedure.

To compute the new code, we compare the subtree code at same level with same root, until we reach level 0. Therefore, the worst case arises when the procedure compares the largest number of subtree codes whose have the largest lenght. The CTree which guarantees it is the complete one in which we add a leaf at the last level on the right side of the CTree. Let l_i be the maximum lenght of a subtree code at level $i = 1, .., L$. So, we have:

$$l_1 = \frac{n-1}{4} \tag{2}$$

$$l_i = \frac{l_{i-1}-1}{3}, \quad i = 2, \ldots, L-1. \tag{3}$$

The root code has lenght equal to n. Since the CTree is complete, the root has 4 subtrees and each of these codes has lenght equal to l_1. The same reasoning holds for l_i: the root subtree code has lenght equal to l_{i-1} and it has 3 subtrees which codes have lenght $(l_{i-1})/3$. The compairison between two code at level i is $O(l_i)$. In the worst case, we have to do two comparaisons at each level except for the level 1 in which they are three. In the worst case therefore the procedure complexity is

$$\begin{aligned}
3l_1 + \sum_{j=2}^{L-1} 2l_j &\leq 3\sum_{j=1}^{L-1} l_j = \left[\frac{n-1}{4} + \frac{\frac{n-1}{4} - 1}{3} + \ldots\right] \\
&\leq \left[\frac{n-1}{4} + \frac{n-1}{4\,3} + \frac{n-1}{4\,3^2} + \ldots\right] \\
&\leq 3\sum_{j=1}^{L-1} \frac{n-1}{4\,3^j} = \frac{3}{4}(n-1)\sum_{j=1}^{L-1}\frac{1}{3^j} \\
&\leq \frac{3}{4}(n-1)\sum_{j=1}^{\infty}\frac{1}{3^j} = \frac{3}{4}(n-1)\frac{3}{2} = \frac{9}{8}(n-1).
\end{aligned}$$

which means that the coding of a new tree is $O(n)$.

The complexity of procedure **Gen** strongly depends on the implementation of GS and its inverse. Since the generation of a new origin or derived trees is $O(n)$, the **Gen** has complexity $O(n^3)$:

- GS function in worst case generates $n-1$ new trees and its code in linear time; so GS is $O(n^2)$ and the size of set D is $O(n)$;
- $InvGS$ function can be carried out in $O(n^2)$ as GS;
- the instructions within while loop are repeated $size(D)$ times; hence the while loop is $O(n^3)$.

A way to improve **Gen** running time is that to define rules to quickly recognize the origin tree with maximum code during $InvGS$ computation.

We now report the general algorithm:

```
procedure Enumeration (n);
begin
   C[1..n]:sets of trees;
   C[1]:=methane; n_of_trees[1..n]:=0; counter:=1;
```

```
   while counter<=n
   do begin
      while C[counter] is not empty
      do begin
         t:=get(C[counter]); Gen(t,counter);
      end;
      counter:=counter+1;
   end;
   output(n_of_trees[n],elapsed_time,C[n]);
end.
```

Let S_i be the number of i-trees, $i = 1, \ldots, n$. Given that each inner while loop is $O(i^3 S_i)$, the complexity of **Enumeration** is given by

$$\sum_{i=1}^{n} i^3 S_i \quad .$$

Given that the following formulas hold true

$$\sum_{i=1}^{n} i^3 S_i \leq \sum_{i=1}^{n} i^3 S_n \leq S_n \sum_{i=1}^{n} i^3 = S_n \left(\sum_{i=1}^{n} i \right)^2 = S_n \frac{n^2(n+1)^2}{4} \quad ,$$

we can state that the following complexity theorem:

Theorem 2 *The AHM procedure has $O(n^4 S_n)$ asymptotic complexity.*

Since also in [19] the generation of a new origin or derived trees is $O(n)$, the HPVK has the same asymptotic complexity. Moreover, the procedure described in Enumeration is the same also for HPVK since it is independent from the molecular code used.

Enumeration is also a general scheme which can be followed in one-to-one enumeration algorithms as for example TNKMS. The complexity of TNKMS is not reported since in [31] is not exatly reported the structure of the algorithm.

Before studying the LKP asymptotic complexity, we give the scheme of the LKP procedure.

```
procedure LKP (n);
begin
   R[1..floor(n/2)]:sets of rooted trees; T: set of trees;
```

```
   R[1]:={0}; n_of_alk:=0; k:=floor(n/2);
   for p = 2 to k do begin            {rooted tree generation}
      R[p]:= emptyset;
      for each [a(x),b(y),c(z)] s.t. condition 1 holds
      do repeat
            put newcode(a(x),b(y),c(z)) in R[p]
         until all_combinations_are_generated;
   end;
   if Odd(n)                            {odd n-tree generation}
   then begin
      for each [a(x),b(y),c(z),d(w)] s.t. condition 2 holds
      do repeat
            put newcode(a(x),b(y),c(z),d(w)) in T;
            n_of_alk:=n_of_alk+1
         until all_combinations_are_generated
   end else begin                      {even n-tree generation}
      for each [a(x),b(y),c(z),d(w)] s.t. condition 3 holds
      do repeat
            put newcode([a(x),b(y),c(z),d(w)]) in T;
            n_of_alk:=n_of_alk+1
         until all_combinations_are_generated
         repeat                  {bicentroids n-tree generation}
            put compose(k(2)) in T;
            n_of_alk:=n_of_alk+1;
         until all_bicentroids_are_generated
   end;
   output(n_of_alk,elapsed_time,T);
end.
```

where: $a(x), b(y), \ldots$ means that we use x a-rooted, y b-rooted and so on, to build one n-tree; *floor* is the $\lfloor\cdot\rfloor$ operator; *newcode* and *compose* return the new code tree. The molecular code is the N-tuple.

The conditions within LKP have been derived from [24] and they are:

Condition 1: $ax + by + cz = \lfloor\frac{n}{2}\rfloor - 1$;
$1 \leq x + y + z \leq 3$ (degree constraint) [4];
$1 \leq a < b < c \leq p - 1$;
$a, b, c \in \{1, \ldots, p-1\}$, $x, y, z \in \{0, .., 3\}$;

Condition 2: $ax + by + cz + dw = n - 1$;

$2 \le x + y + z + w \le 4$ (degree constraint);

$1 \le a < b < c < d \le k$ (Jordan's th.);

$a, b, c, d \in \{1, \ldots, k\}$, $x, y, z, w \in \{0, .., 4\}$;

Condition 3: $ax + by + cz + dw = n - 1$;

$3 \le x + y + z + w \le 4$ (degree constraint);

$1 \le a < b < c < d < k$ (Jordan's th.);

$a, b, c, d \in \{1, \ldots, k - 1\}$, $x, y, z, w \in \{0, .., 4\}$

The above conditions state the rules which have to holds while generating the rooted trees, odd and even n-trees, respectively.

Let R_i be the cardinality of i-rooted set. Two lemmas follows:

Lemma 1 *Given i, the number of possible combinations with m i-rooted trees are $O(R_i^m)$. The number of possible combinations with m i-rooted trees, $i := 1, .., \lfloor \frac{n}{2} \rfloor$, are $O((max_i R_i)^m)$.*

Lemma 2 *The number of triples which hold condition 1 are $O(n^2)$. The number of quadruples which hold conditions 2 and 3 are $O(n^3)$.*

Proof: *Let k be equal to $\lfloor \frac{n}{2} \rfloor - 1$. The triples which hold condition 1 are equivalent to all triple of number (α, β, γ) which sum are equal to k and $\alpha, \beta, \gamma \in \{0, \ldots, k\}$. Fixing α, the problem reduces to counts the couples (β, γ) s.t. $\beta + \gamma = k - \alpha$ and $\beta, \gamma \in \{0, \ldots, k - \alpha\}$. These couples are $\lceil \frac{k-\alpha+1}{2} \rceil$. Therefore, all triples T are given by*

$$T = \sum_{\alpha=0}^{k} \left\lceil \frac{k - \alpha + 1}{2} \right\rceil . \tag{4}$$

which is equivalent to

$$T = \begin{cases} 2 \sum_{j=1}^{\lceil \frac{k}{2} \rceil} j & \textit{for odd } k \\ \frac{k}{2} + 2 \sum_{j=1}^{\lceil \frac{k}{2} \rceil - 1} j & \textit{for even } k \end{cases} . \tag{5}$$

Both for odd and even k, T is $O(n^2)$. The quadruple which hold conditions 2 and 3 are $O(n^3)$: fixing the first element, we can associate to it $O(n^2)$ triples.

These lemmas imply that:

- the rooted tree generation is $O(n^3 R^3_{\lfloor \frac{n}{2} \rfloor})$;
- the bicentroids n-tree generation is $O(n R^2_{\lfloor \frac{n}{2} \rfloor})$ (compose is a linear function);
- the n-tree generation is $O(n^3 R^4_{\lfloor \frac{n}{2} \rfloor})$.

Therefore, we can state the following complexity theorem:

Theorem 3 *The LKP procedure has $O(n^3 R^4_{\lfloor \frac{n}{2} \rfloor})$ asymptotic complexity.*

Finally, Table III summarize our asymptotic complexity analysis.

5 Running time comparaisons

The concern of this section is to provide an analysis of the running time experiences made by the computer programs implementing the algorithms previously presented.

In Table II we report the comparaison between the application of the Reverse Search and the use of an unordered list to check the uniqueness of the derived trees. These results show how the Reverse Search is better than the unordered list when n increases. This means that the generation of all origin trees is more efficient than comparing the derived tree with the ones previously generated. Moreover, we save memory space since the Reverse Search do not require to memorize all the generated codes.

Table III reports the running times of the presented algorithms [3]. The computer programs are implemented in Pascal with the exception of TNKMS for which we used the Fortran77 program reported in [31]. Moreover, for large n, the list can exceed the total memory available. On the contrary, Reverse Search requires less memory space than a list. Therefore, the Reverse Search allow to enumerate alkanes with greater n than list.

TMKMS appears as the worst algorithm to produce the alkane molecular family. Moreover, TNKMS generates more tree classes than the other ones. However the great differences in running time between the other algorithms can balance the larger output produced by TNKMS. In fact TNKMS does not provide rules which guarantee uniqueness.

AHM and HPVK have similar running time and their gaps are caused only for different implementations.

LKP is surprinsingly the better algorithm. For $n = 21$ LKP is about four times faster than HPVK, about five times faster than AHM and about 1320 times faster than TNKMS.

The main reason of this good performance is essentially the fact that LKP builds directly the n-trees from the $\lfloor \frac{n}{2} \rfloor$-rooted trees while other algorithms have to produce all set of j-trees, $j = 1, \ldots, n-1$. This means that to produce the 366319 20-trees, LKP has to generate 879 rooted trees while AHM, HPVK and TNKMS have to generate 251731 trees.

Although the AHM, HPVK and TNKMS algorithms return larger outputs than LKP, that is they return the alkanes molecular family up to n. We suppose that also LKP computes them. An upper approximation of its running time is given summing up the running times between 1 and n reported in Table III. This running time approximation is less than running time of HPVK or AHM; for example, with $n = 24$ LKP saves about 14 minutes. Moreover, an optimized LKP procedure which computes the molecular family up to n computes once the rooted trees required rather than n times as in the upper approximation.

We can conclude that LKP seems the best algorithm to generate the alkane molecular family.

6 Conclusions

In this paper we presented four alkane molecular family enumeration algorithms and compared their efficiency both theoretically and experimentally.

The key issue of any enumeration algorithm is the avoidance of duplications during the generation.

The one-to-one enumeration algorithms require some rules which avoid duplicates. We have shown how Reverse Search and Symmetry technique are both useful avoiding duplicates and saving memory space. Moreover, they speed up the running time of the algorithm. The AHM and the HPVK algorithms use these rules successfully and they require less memory space and running time than the TNKMS one.

The Jordan's Theorem guarantees the uniqueness in the constructive enumeration algorithm. As shown in the previous section, LKP is surprinsingly the better algorithm.

Remarkable works about enumeration by researchers not deeply involved in algorithmical issues usually avoid to study asymptotic complexity and running time behaviour.

Asymptotic complexity studies allow to guess the algorithm behaviour: algorithms with low asymptotic complexity are almost always faster than algorithms with higher complexity. This is the case of LKP with respect to AHM and HPVK.

Running time analysis (on the same computer machine) are necessary to understand which is the better algorithm also from the practical point of view. This is the case of AHM and HPVK, for example.

Therefore, we want to stress the importance of this kind of studies developing good algorithms.

An interesting future investigation could be the extension of LKP algorithm in order to make it a complete tool for generate trees. The first step is to make it more efficient as shown in section 5. Then LKP could be extended in order to compute a larger family of trees, that is trees with n larger than 4. LKP can be further extended following the statements within the works of Kvasnička and Pospichal in [24] [25].

References

[1] The node "center" of a tree is defined as the remaining node of the recursive procedure which eliminates the leafs of the tree at each step. As stated in Jordan's theorem, we can have one node as center - centroid - or two nodes as centers - bicentroid.

[2] These algorithms are developed in parallell in May 1995 when the first two authours are visiting at EPFL of Lausanne.

[3] The running time experiences are made on Sylicon Graphics Indingo 4000, processor Risc IP20 100 MHz, 48 Mbytes of memory.

[4] Since we have to build 4-degree constrained tree, each rooted tree can have more than 3 subtrees. Otherwise, if the rooted trees has 4 or more subtrees give raise to a 5 or more degree constrained tree.

[5] Abe H.; Okuyama T.; Fujiwara I.; Sasaki S., A Computer Program for Generation of Constitutionally Isomeric Structural Formulas. *J.Chem.Inf.Comput.Sci.* **1984**, *24*, 220-229.

[6] Aringhieri R., Metodi dell'Ottimizzazione Combinatoria applicati alla Chimica: L'Enumerazione ed il Calcolo del Numero di Hyper Wiener di Molecole di Alcano. *Tesi di laurea*, **1996**, Dipartimento di Informatica, Università di Pisa.

[7] Avis D.; Fukuda K., A Pivoting Algorithm for Convex Hulls and Vertex Enumeration of Arrangements and Polyhedra. *Discrete Comput. Geometry* **1992**, *8*, 295-313.

[8] Avis D.; Fukuda K., Reverse Search for enumeration. *Discrete Applied Mathematics* **1996**, *65*, 21-46.

[9] Balaban T. S.; Filip P. A.; Ivanciuc O., Computer Generation of Acyclic Graphs based on Local Vertex Invariants and Topological Indices. Derived Canonical Labelling and Coding of trees and Alkanes. *J.Math.Chem.* **1992**, *11*, 79-105.

[10] Bangov I. P., Computer-Assisted Structure Generation from a Gross Formula 3. Alleviation of the Combinatorial Problem. *J.Chem.Inf.Comput.Sci.* **1990**, *30*, 277-289.

[11] Berzelius J. J., *Ann. Phys. Chem.* **1830**, *19*, 305-.

[12] Cayley A., On the Theory of Analytical Forms called Trees. *Philos. Mag.* **1857**, *13*, 172-176.

[13] Cayley A., On the Mathematical Theory of Isomers. *Philos. Mag.* **1874**, *47*, 444-446.

[14] Cayley A., On the Analytical Figures called Trees in Mathematics and Their Applications to the Theory of Chemical Combinations. *Ber. Dtsch. Chem. Ges.* **1875**, *8*, 1056-1059.

[15] Contreras M. L.; Valdivia R.; Rozas R., Exhaustive Generation of Organic Isomers. 1. Acyclic Structures. *J.Chem.Inf.Comput.Sci.* **1992**, *32*, 323-330.

[16] Corey E. J.; Wipke W. T., *Science* **1969**, *166*, 178-.

[17] Corey E. J., *Quart. Rev. Chem. Soc.* **1971**, *25*, 455-.

[18] Hansen P.; Jaumard B.; Labatteux C.; Zeng M., Coding Chemical Trees with the Centered N-Tuple Code. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 782-790.

[19] Hansen P.; Pitteloud G.; Van der Klink R., Enumeration d'Alcanes par le Code N-Tuple et la Recherche Inversee. *Travail de semestre* 1995, DMA, EPFL, Lausanne.

[20] Hendrickson J. B.; Parks C. A., Generation and Enumeration of Carbon Skeletons. *J.Chem.Inf.Comput.Sci.* **1991**, *31*, 101-107.

[21] Jackson M. D.; Bieber T. I., Applications of Degree Distribution. 2. Construction and Enumeration of Isomers in the Alkane Series. *J.Chem.Inf.Comput.Sci.* **1993**, *33*, 701-708.

[22] Jordan C., *Reine Angew.Math.* **1869**, 185-.

[23] Kvasnička V.; Pospichal J., Canonical Indexing and Constructive Enumeration of Molecular Graphs. *J.Chem.Inf.Comput.Sci.* **1991**, *56*, 1777-1802.

[24] Kvasnička V.; Pospichal J., Constructive Enumeration of Acyclic Molecules. *Collect.Czech.Chem.Commun.* **1991**, *56*, 1777-1802.

[25] Kvasnička V.; Pospichal J., An Alternative Approach for Constructive Enumeration of Graphs. *Collect.Czech.Chem.Commun.* **1993**, *58*, 754-774.

[26] Lederberg J.; Sutherland J. L.; Buchanan B. G.; Feigenbaum E. A.; Robertson A. V.; Duffield A. M.; Djerassi C. J., *J. Am. Chem. Soc.* **1969**, *91*, 2973-.

[27] Lederberg J.; Sutherland J. L.; Buchanan B. G.; Feigenbaum E. A.; Lindsay R. K. , Application of Artificial Intelligence for Organic Chemistry: the DENDRAL project. McGraw-Hill: New York, **1980**.

[28] Polya G.; Read R.C.; in Combinatorial Enumeration of Graphs, Groups, and Chemical Compounds. Springer-Verlag: New York, **1987**; p 58-74.

[29] Read R. C., The Coding of Various Kind of Unlabelled Trees in *Graph Theory and Computing*; Ed Read R. C., Academic Press: New York, **1972**, Chapter 12, p 153.

[30] Read R. C., A New System for Designation of Chemical Compounds. 1. Theoretical Preliminaries and Coding of Acyclic Compounds. *J.Chem.Inf.Comput.Sci.* **1093**, *23*, 135-182.

[31] Trinajstić N.; Nicolić S.; Knop J. V.; Müller W. R.; Szymanski K., Computational Chemical Graph Theory. Ellis Horwood: New York. **1991**.

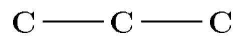

Figure 1: *propane* C_3H_8.

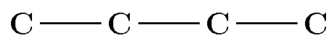

Figure 2: *nbutane* C_4H_{10}.

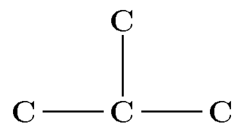

Figure 3: *isobutane* C_4H_{10}.

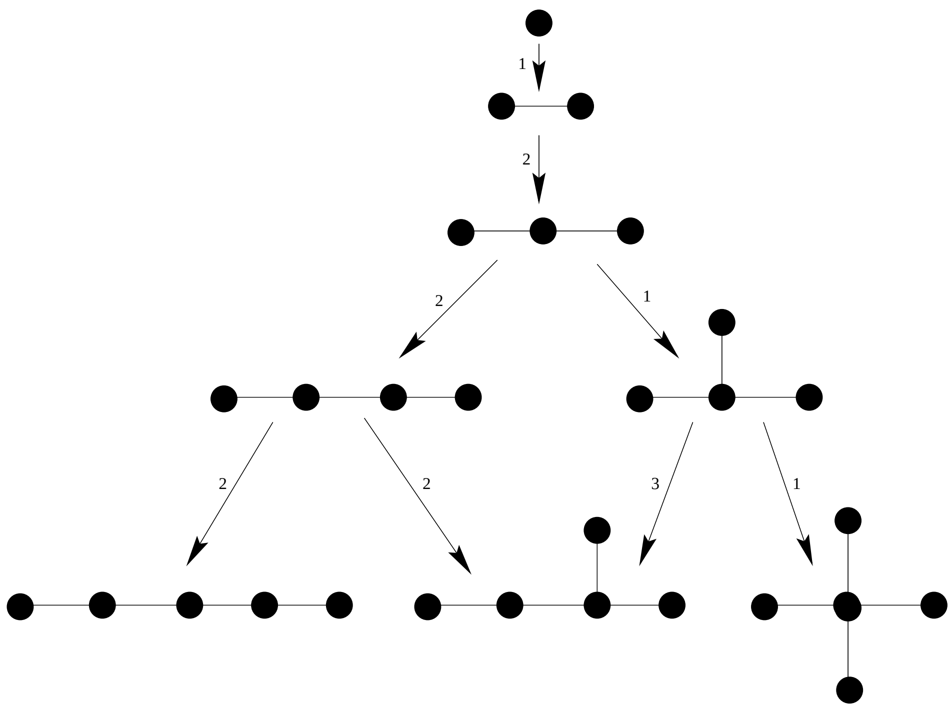

Figure 4: *One-to-one algorithm example.*

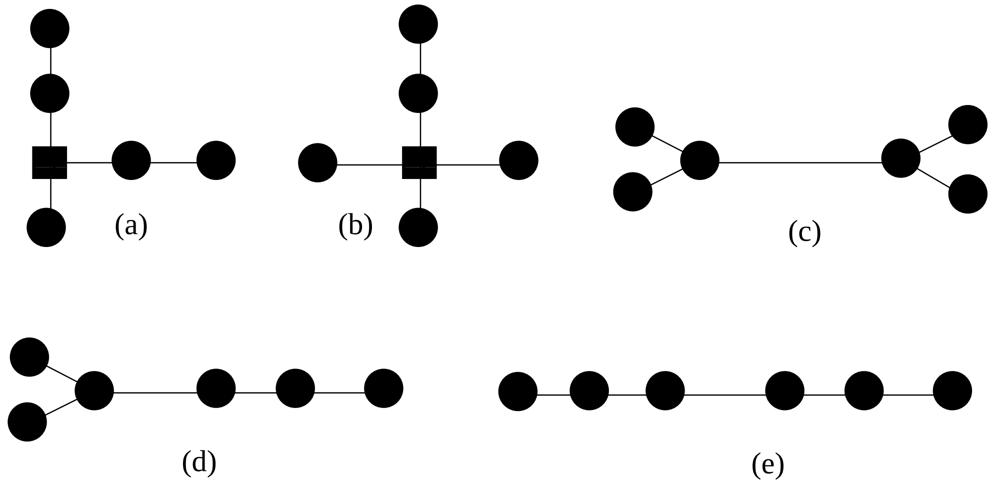

Figure 5: *Constructive algorithm example.*

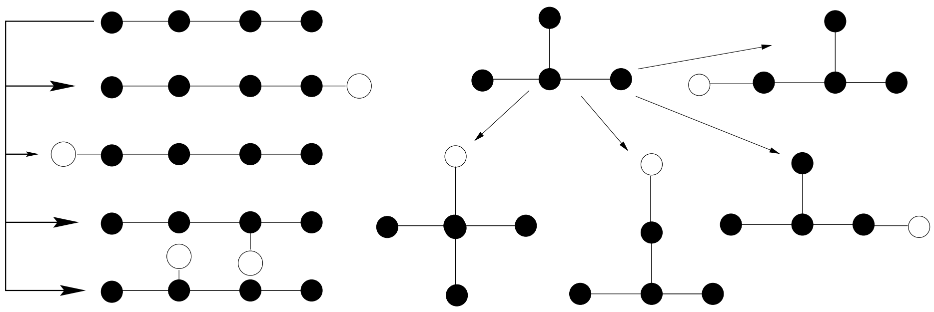

Figure 6: *Tree symmetrie.*

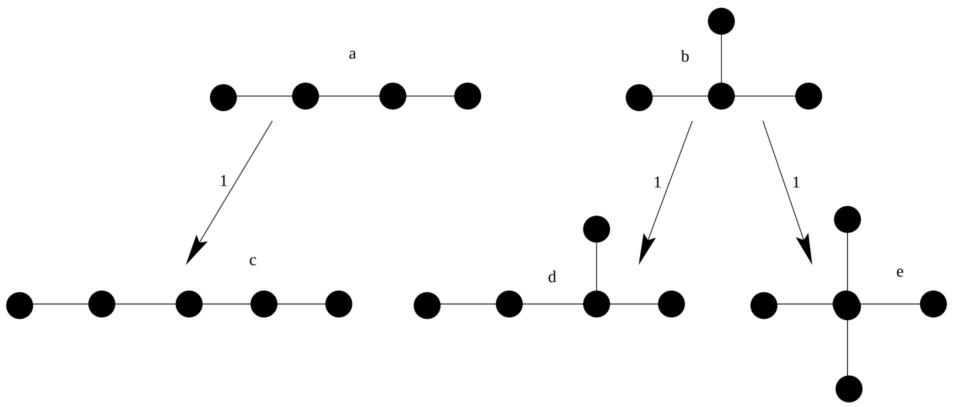

Figure 7: *Application of Reverse Search and Symmetries techniques.*

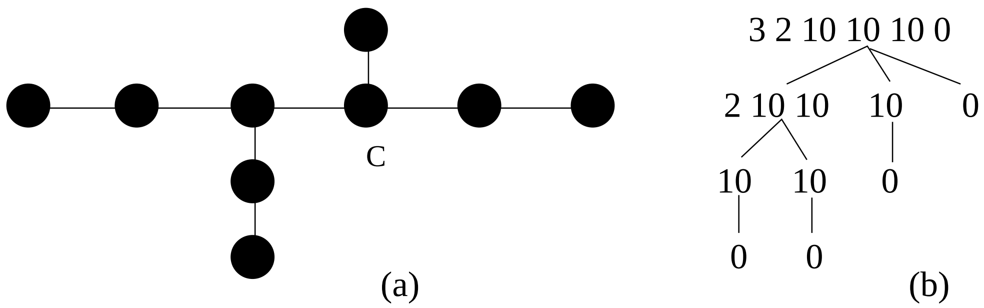

Figure 8: *CTree example.*

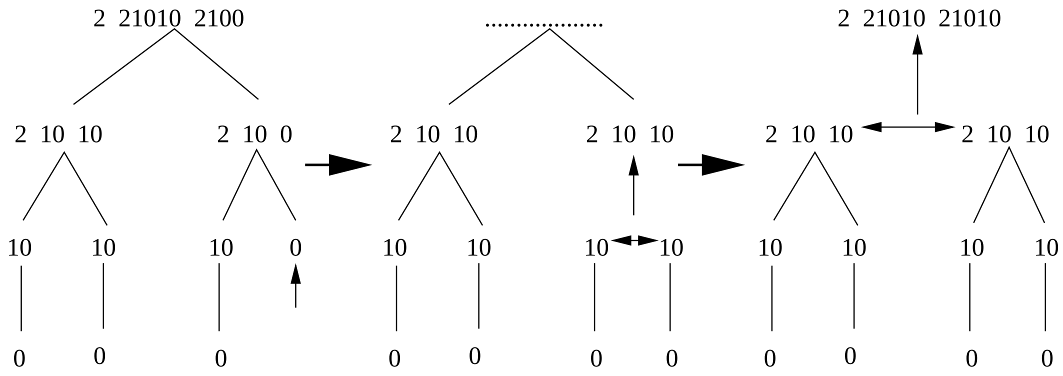

Figure 9: *Recoding Rule example.*

Table I: Asymptotic complexities			
LKP	**HPVK**	**AHM**	**TNKMS**
$O(n^3 R^4_{\lfloor \frac{n}{2} \rfloor})$	$O(n^4 S_n)$	$O(n^4 S_n)$	- -

Table II: Reverse Search versus List (times in ms)		
N	**Unordered List**	**Reverse Search**
10	40	80
11	140	220
12	620	580
13	3140	1550
14	23150	4230
15	175340	12210

Table III: Running time of the algorithms					
N	n. of trees	**LKP**	**HPVK**	**AHM**	**TNKMS**
16	10359	0”	0”	0”	45”
17	24894	0”	2”	2”	2’43”
18	60523	2”	6”	7”	9’50”
19	148284	4”	14”	16”	35’44”
20	366319	15”	37”	42”	2h11’20”
21	910726	22”	1’38”	1’52”	8h04’39”
22	2278658	1’34”	4’17”	4’57”	- -
23	5731580	2’25”	11’11”	13’09”	- -
24	14490245	10’18”	29’22”	31’39”	- -
25	36797588	15’58”	1h18’43”	- -	- -
26	93839412	1h06’44”	3h33’50”	- -	- -
27	240215803	1h46’25”	9h09’35”	- -	- -