Journal of Mathematical Chemistry 11(1992)187–197

CODING AND ENUMERATION OF TREES THAT CAN BE LAID UPON A HEXAGON LATTICE

E.C. KIRBY
Resource Use Institute, 14 Lower Oakfield, Pitlochry, Perthshire PH16 5DS, Scotland, UK

Abstract

A tree that can be superimposed upon a hexagon lattice is called a hexagon lattice tree. A method for mechanically coding, enumerating and drawing these objects is described, and has been tested for trees with up to ten vertices. For storage and information transmission, the code uses an expanded version of the N-tuple code in which edge vector elements having one of four possible values are inserted. For establishing uniqueness, it is used in combination with a hexagon lattice reference grid whose vertices are numbered sequentially in the tightest possible outward spiral. Published rules for the derivation of N-tuple codes by hand are commented on, and a small error pointed out.

1. Introduction

In recent work on Hamiltonian or path-Hamiltonian polyhexes [1,2] and sextet 2-factorable polyhexes [3], a special subgraph of a polyhex referred to as a "branching graph" was used. (A Hamiltonian path – a path that visits every vertex just once – is a particular kind of spanning tree, and is of chemical interest in connection with magnetic ring current calculations [4,5].) The branching graph of a polycyclic graph is the set of vertices of degree greater than two with the set of edges joining pairs of these branching vertices. Only a little has been done towards the characterisation of such objects [6], and this work was motivated partly in an attempt to understand them better. For example, the branching graphs of some polyhexes are trees, but which trees and why?

Several polyhexes can share a tree as their branching graph, by the tree assuming different geometric conformations. We thus wanted to know all the ways in which a given tree (of maximum valency three) can be laid upon a hexagon lattice. Trees that can be so placed are called lattice trees, a class of lattice animal, and they have been used to represent randomly branched polymers in dilute solution. They can also model geometric isomers of hydrocarbon polyenes, and similar planar structures in which rotation is restricted.

Another practical subject in which the sets of such isomers are of interest is nutrition and fatty acid chemistry for, although little attention has been paid to this point, there have been suggestions that the geometric isomers of triglycerides arising

in partial hydrogenation processes may be metabolised in different ways that are not all good for health [7].

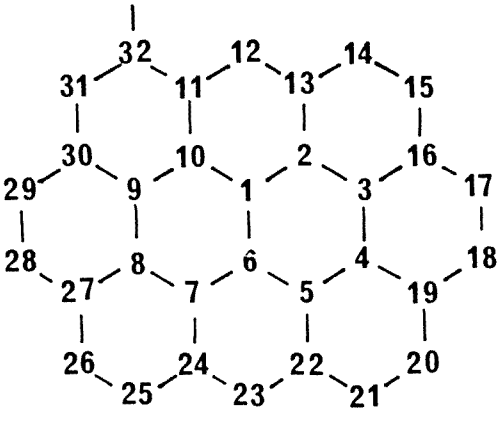
Attempts to derive mathematical expressions for deducing the numbers of lattice animals appear to have been only partially successful [8]. In any case, for the purposes mentioned here the ability to draw the whole set, not just to determine their total number, is required. The strategy has been to develop a moderately simple method for encryption of the isomers that can reliably be used to generate first a field of code values containing all those of interest, and then a list of all distinct isomers.

2. The coding of hexagon lattice trees

Amongst all the possible trees, only 3-trees are of interest here, because only they can be hexagon lattice animals. The first problem is how best to encode such animals concisely, and to recognize any previously encountered.

There are twelve possible ways of drawing a 3-tree so that its edges are parallel to the edges of a hexagon lattice: three rotational variants and their mirror images, for two classes of vertex (corresponding to whether, when acting as a centre of rotation on a vertically aligned lattice, the vertex can have an incident vertical edge "up" or "down"). To deal with these systematically, it is useful to number the hexagon lattice in a spiral manner.

Such a spiral grid was shown earlier [3] to provide a convenient means of encoding structures, such as polyhexes, that are made of hexagons and are superimposable upon a hexagon lattice. The same principle can be applied here, by using a grid in which the vertices (rather than, as previously [3], hexagons) are numbered sequentially in the tightest possible outward spiral, arbitrarily chosen to be clockwise (**1**). It is necessary to examine the rotations about each vertex in turn.



(1)

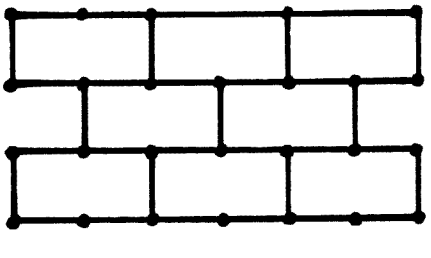
The total number of superimpositions to be made is $6 \times N$ (including possible symmetry-induced redundancies). These are listed, and the one yielding the lowest

lexicographically ordered set of vertex reference numbers is chosen. An edge list is then written in terms of pairs of these vertex reference numbers, and this defines the lattice tree.

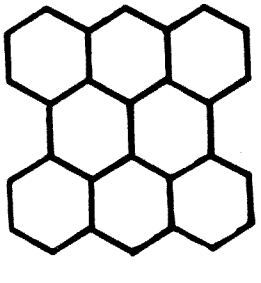
This list could itself be used as a code for the lattice tree but, as with edge lists generally, it is somewhat inconvenient for manipulation, and it tends to have rather large vertex label number values. A further consideration is that this code, whilst it provides a unique identification and allows different lattice animals to be distinguished, does not help towards their systematic *enumeration.*

Trees in general, with undefined geometry, can be encoded uniquely and concisely with the *N*-tuple code of Knop and co-workers [9]. (See appendix 1 for a description, and for a report on a small error that has appeared in more recently published work.) This can be expanded to provide a concise means of conveying geometric information too.

To do this, the *N*-tuple code of n digits for a 3-tree is expanded into what is here called a "*GN*-tuple" code ("*G*" signifying "geometric") with $2 \times n - 1$ digits. The extra $n - 1$ elements are vector digits, and are interspersed between each pair of the n *N*-tuple digits. They indicate in which of four possible directions an edge from a lower to a higher numbered vertex should be drawn. Each such *GN*-tuple refers unambiguously to a specific lattice animal. Reconstruction will reproduce the animal on a rectangularly distorted hexagon lattice (**2**) (see ref. [9]) rather than the more familiar one (**3**), but this is easily corrected. It is also fairly



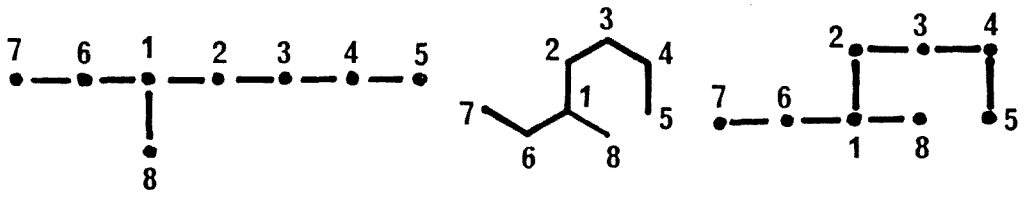
(2)



(3)

simple to write a program to read a *GN*-tuple code and to generate a recognizable screen representation of it. This use of the spaces between the elements of the *N*-tuple code was recently foreshadowed in ref. [10], where it is pointed out that non-carbon structures, and multiple bonds, can be accommodated in this way.

As an example, one of the fifteen possible conformations of the tree (**4**) is (**5a**) or (**5b**) (they are equivalent). Its *N*-tuple code value (appendix 1) is 31110100, which forces the labelling shown.



N-tuple code = 31110100

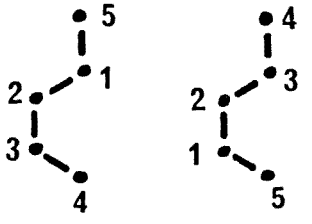
(4) (5a) (5b)

For the eight-vertex tree (4) there are of course seven edges. These can be listed together with a mapping to elements of the N-tuple, and assigned vectors as follows:

Elements of N-tuple code	Edge	Edge vector (1 = up; 2 = right; 3 = down; 4 = left)
3	null	null
1	1–2	1
1	2–3	2
1	3–4	2
0	4–5	3
1	1–6	4
0	6–7	4
0	1–8	2

Interleaving the edge vectors with the elements of the N-tuple code gives a GN-tuple code value of 311212130414020 for (**5a**)/(**5b**).

This provides an alternative means for unambiguously describing a particular hexagon lattice animal. In comparison with the spiral grid edge-list described above, this GN-tuple code is very convenient for manipulation, but it is somewhat more difficult to ensure uniqueness. This is because, although there are only twelve possible orientations (and the position of the centre of rotation makes no difference), the code value also depends upon how the tree is labelled. The labelling forced upon the tree by its N-tuple code is unique only to the point of isomorphism. Vertices that are equivalent by symmetry in a tree of undefined geometry may no longer be so in one of the tree's lattice animals; for example, (**6a**) and (**6b**) have different



(6a) (6b)

vector tuples (4321 and 1212, respectively). It would be possible to ensure that every possible labelling is considered, but this seems quite complicated.

The *GN*-tuple does, however, have another very attractive feature that makes its use worthwhile: it provides a simple means of describing every possible hexagon lattice tree at least once. Thus, the *GN*-tuple of an n-vertex tree will contain $n-1$ elements, each of which has a value in the range 1–4 or, in other words, the geometry can be described by a base-4 number containing $n-1$ digits. A set of $4^{(n-1)}$ base-4 numbers will therefore contain at least one representation of each possible conformation, and provided that repeated structures can be recognised, enumeration is straightforward.

The remaining problem is to ensure that repeated structures *are* recognised, so that unique *GB*-tuple codes can be generated, and for this work, this was achieved by combining the two approaches of a "spiral" edge-list and a "geometric tuple" into a single encoding and enumeration algorithm.

3. The coding and enumeration of the hexagon lattice 3-trees

A given tree of n vertices has one unique N-tuple code [9] and a number of ways in which it can be superimposed upon a hexagon lattice. The working enumeration variable is the tuple of $n-1$ vector digits that represent the orientation of the edges. Every array of digits from 111 . . . 1 to 444 . . . 4 is tested to see whether a lattice tree can be drawn from it. If it can, then the vector tuple is used to generate a subset of the vertices of the hexagon reference grid (**1**), and its minimum rotational variant is selected for temporary storage. In most cases, vertices alone are sufficient for this purpose, but it is of course sometimes necessary to invoke the connections too.

If the same structure is generated by some other set of digits (corresponding to another implicit N-tuple labelling), it will be recognised by its spiral grid synonym, and the new set of vector digits will be substituted if and only if lexicographically smaller than the existing one in store. The end result is thus a list consisting of *GN*-tuple codes defined in a unique manner representing all the possible hexagon lattice 3-trees with n vertices. It should be emphasised that in this scheme the value of the *GN*-tuple code does not necessarily have its minimum possible value; rather, it has the minimum value associated with an edge list written in terms of the set of minimum vertex values on a spirally numbered hexagon reference grid.

A simple example is shown in appendix 2, and some numerical results of the enumeration are shown in table 1. The time taken increases roughly with the fourth power of $n-1$, and while this was satisfactory for $n \leq 10$, a more sophisticated method would soon be needed for larger systems. A small economy can be made by standardising the first two elements of the trial vector tuple. This is because there is only one non-equivalent way in which two incident edges can be drawn upon a hexagon lattice; so search time need be proportional only to $(n-3)^4$. It is likely that, with care, some further standardisations of this nature could be made before they

Table 1

Numbers of geometric isomers of trees of up to degree three (hexagonal lattice animals) with up to ten vertices. N = number of vertices; R = sequential reference number in ascending order of N-tuple (Knop et al.); NT = N-tuple (ref. [9]); I = number of geometric isomers counted; $B1$ = number of geometric isomers that are the branching graph of some polyhex; $B2$ = number of geometric isomers that are the branching graph of some singly connected poly-polyhex [6].

N	R	NT	I	$B1$	$B2$	N	R	NT	I	$B1$	$B2$
1	1	0	1			10	1	2111111100	61	1	15
2	1	00	1	1	0		2	3110110110	10	1	0
3	1	200	1				3	3111011010	54	0	4
4	1	2100	2	1	0		4	3111011100	31	1	1
	2	3000	1	1	0		5	3111101010	26	0	3
5	1	21100	3				6	3111101100	52	3	6
	2	31000	1				7	3111110100	54	1	7
6	1	211100	6	1	2		8	3111111000	27	1	7
	2	310100	3	1	0		9	3111120000	5	1	2
	3	311000	2	1	0		10	3111210000	12	1	1
	4	320000	1	1	0		11	3112100100	16	1	1
7	1	2111100	9				12	3112101000	6	0	1
	2	3101010	2				13	3112110000	14	0	0
	3	3110100	8				14	3121010100	14	0	0
	4	3111000	4				15	3121100100	27	3	0
	5	3120000	1				16	3121101000	14	0	0
	6	3210000	2				17	3121110000	14	1	1
8	1	21111100	18	1	5		18	3121200000	2	1	0
	2	31101010	8	0	1		19	3210101010	7	0	1
	3	31101100	9	2	0		20	3211001010	14	1	0
	4	31110100	15	1	1		21	3211001100	18	2	0
	5	31111000	7	1	2		22	3211010100	30	0	1
	6	31120000	2	0	0		23	3211011000	9	0	0
	7	31210000	4	1	0		24	3211100100	27	1	0
	8	32100100	6	1	0		25	3211101000	13	1	1
	9	32101000	3	1	0		26	3211110000	13	1	4
	10	32110000	4	1	0		27	3211200000	3	0	0
	11	32200000	1	1	0		28	3212000100	7	1	0
9	1	211111100	31				29	3212001000	4	0	0
	2	311011010	15				30	3212100000	7	1	0
	3	311101010	14				31	3220011000	2	0	0
	4	311101100	29				32	3220020000	1	1	0
	5	311110100	27				33	3221000100	9	0	0
	6	311111000	14				34	3221001000	8	1	0
	7	311120000	2				35	3221010000	4	0	0
	8	311210000	7				36	3221100000	7	2	0
	9	312100100	9				37	3222000000	2	1	0
	10	312101000	4								
	11	312110000	7								
	12	321010100	8								
	13	321100100	15								
	14	321101000	8								
	15	321110000	7								
	16	321200000	2								
	17	322001000	1								
	18	322100000	4								

become too unwieldy, but these were not attempted here. In table 1, the set of each tree size is ordered by N-tuple code. If the numbers of isomers are plotted against their ordinal number with n increasing (not shown here), a somewhat similar overall distribution pattern occurs in each case, though with increasing complexity. It is possible that this may be fractal in type.

4. The enumeration of hexagon lattice paths

This special case of the preceding problem can be solved independently by using the nomenclature of organic chemistry for unsaturated hydrocarbon chains, where each double bond that is not at the end of a chain is described as "cis" or "trans". Thus, a chain of m edges (and $m + 1$ vertices) is treated as a fully conjugated polyene, so that each of the $m - 2$ internal edges may be categorised as cis or trans, and the whole uniquely encoded as a sequence of elements having one of two values. Obviously, this is equivalent to a binary number, where bits 0 and 1 are used to represent (in an arbitrary mapping) cis and trans, respectively. A set of binary numbers of X digits contains codes for all the cis–trans isomers (hexagon lattice chains) with $X + 3$ vertices. It remains to identify those that are iso-codal. The numbers can be partitioned into those that are palindromic (i.e. that read the same forwards as backwards and have a vertical plane of symmetry) and those that are not. The former represent isomers, while the latter can be grouped into pairs that are backward reading images of each other. One more restriction can be imposed: it can easily be seen that in planar hexagon lattice animals, it is not impossible to have a continuous sequence of more than three "cis" edges. Using the above convention therefore, all binary numbers containing strings of more than three zeros are eliminated.

For example: a chain with seven vertices has six edges and four internal edges. The field to be examined for cis–trans isomers therefore has sixteen binary numbers. If 0 represents cis and 1 trans, then the list is as follows:

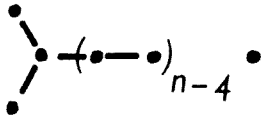
1.		0000	Not valid (more than 3 consecutive cis edges)
2.	1	0001	Paired
3.	2	0010	Paired
4.	3	0011	Paired
5.		0100	The same as 3
6.	4	0101	Paired
7.	5	0110	Symmetrical
8.	6	0111	Paired
9.		1000	The same as 2
10.	7	1001	Symmetrical
11.		1010	The same as 6
12.	8	1011	Paired
13.		1100	The same as 4
14.		1101	The same as 12
15.		1110	The same as 8
16.	9	1111	Symmetrical

Table 2

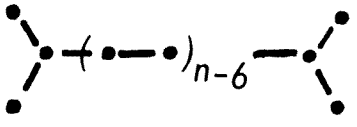
A count of the possible cis–trans isomers of fully conjugated polyenes deduced from binary numbers.

Number of vertices	Number of isomers
2	1
3	1
4	2
5	3
6	6
7	9
8	18
9	31
10	61
11	110
12	214
13	398
14	771
15	1458

There are therefore nine lattice chains with seven vertices, and they can easily be drawn from these binary codes. Table 2 shows some more results of this method of enumeration, and where the results overlap with table 1, there is full agreement. It is possible to extend this approach to some other fairly simple trees such as (7) and (8). In each case, it is necessary to determine whether there are symmetry



(7)



(8)

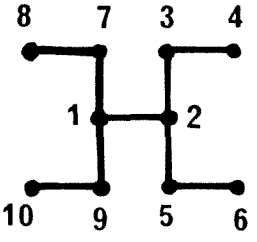
redundancies, and what continuous cis sequences are not allowed. Not surprisingly, however, the process soon becomes very complicated with increasing size and number of branches.

Appendix 1: *N*-tuple codes

This efficient code [9] is applicable to trees. The *N*-tuple code value for a tree with *n* vertices consists of a string of *n* digits, the last one of which is always zero. The first has a value corresponding to the highest valency within the graph, and if more than one vertex has this valency, then the root vertex will be the one that results in the lexicographically highest code value. The next digit represents the

adjacent vertex of highest valency, but its actual value is less than this. Again, if there is a choice then each possibility is followed until a difference emerges, whereupon the highest valued route is taken. The process continues in a "depth first manner" until all vertices of the tree have been accounted for, and the highest N-tuple code value obtained.

Each element of the N-tuple code is associated with a particular vertex, and so there is an induced labelling of the graph. The tree represented by an N-tuple string is unique, in the sense that no pair of non-isomorphic trees can have the same value. Where there is symmetry, however, the induced *labelling* is not unique, and the tree (**9**), for example, could be relabelled in several equally legitimate ways.



N-tuple code = 3210101010

(**9**)

Computer algorithms have been written for generating N-tuple codes, and these have been used for enumerating trees [9]. Subsequently, the technique was used as part of a more comprehensive code that can be used for polycylic structures [11–13]. It should be noted, however, that a minor, though slightly misleading error has arisen during this work. In an effort to simplify code derivation, Randić and others formulated a set of rules suitable for paper-and-pencil encryption. These are:

(1) Locate vertices of the highest valency.

(2) Locate the longest possible path.

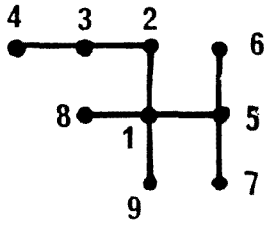
(3) Backtrack to the last branching point to visit all the vertices in that branch.

(4) Continue the process until all vertices branching from the longest path have been accounted for.

(5) Locate the next longest path and continue the process until all vertices in all paths have been recorded.

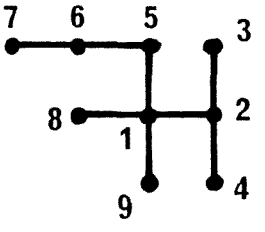
However, step 2 (and therefore also step 5) is not universally applicable. Thus, the code for (**10**) derived by this method is 411020000, yet the lexicographically largest possible (and therefore correct) code is 420011000.

By agreement with one of the authors [14], the following amended version of step 2 is suggested:



Incorrect N-tuple code: 411020000

(10a)



Correct N-tuple code: 420011000

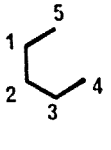
(10b)

Locate the path that includes the earliest visits to the largest number of branching vertices, with the longest such path being chosen if there is more than one, or if only non-branching paths are available.

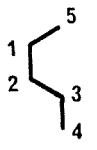
Appendix 2: The search for hexagon lattice trees of the 5-chain as an example of encoding and enumeration with the *GN*-tuple code and a spirally numbered hexagon reference grid

The 5-chain has four edges, so the tuples from 3211 to 4444 (each digit >0 and <5) are examined in combination with the N-tuple code for the 5-chain (21100), to see whether a lattice animal can be drawn. If it can, then its covering of spiral reference numbers is minimised as described in the text. This generates the following list:

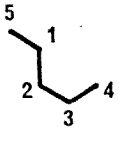
	Trial vector tuple	Struct. ref.	Minimised set of numbers covered on hexagon grid (1)	Corresponding edge list	Corresponding vector tuple	Struct. ref.
1	3222	(11)	1 2 3 4 5	1–2 2– 3 3–4 4– 5	2344	(15)
2	3232	(12)	1 2 3 4 10	1–2 1–10 2–3 3– 4	4443	(16)
3	3224	(13)	1 2 3 4 10	1–2 1–10 2–3 3– 4	2234	(17)
4	3234	(14)	1 2 3 9 10	1–2 1–10 2–3 9–10	2224	(18)



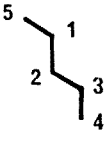
(11)



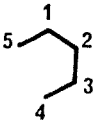
(12)



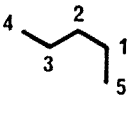
(13)



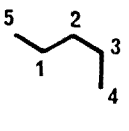
(14)



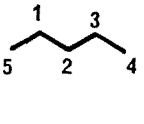
(15)



(16)



(17)



(18)

Only single tuples for 1 and 4 are found. There is another lower valued tuple that is possible for number 1, namely 1444, obtained by reversing label pairs 1,5 and 3,4 in (**15**), but this is ignored in the algorithm used because, to save time, testing starts with the tuple 3211. The third structure is found from its edge list to be identical to number 2, but its lower valued vector tuple (2234) is selected to form the final *GN*-tuple code, 221213040. The search therefore yields three distinct lattice paths, and these are coded as 221314040, 221213040 and 221212040, corresponding to (**15**), (**17**) and (**18**), respectively.

References

[1] E.C. Kirby, J. Math. Chem. 4(1990)31.
[2] E.C. Kirby, *Proc. MATH/CHEM/COMP Conf.*, Dubrovnik (1990), J. Math. Chem. 8(1991)77–87.
[3] E.C. Kirby, J. Chem. Soc. Faraday Trans. 86(1990)447.
[4] I. Gutman, R.B. Mallion and J.W. Essam, Mol. Phys. 50(1983)859.
[5] B. O'Leary and R.B. Mallion, in: *Graph Theory and Topology in Chemistry*, ed. R.B. King and D.H. Rouvray, Studies in Physical and Theoretical Chemistry 51 (Elsevier, Amsterdam, 1987), pp. 544–551.
[6] I. Gutman and E.C. Kirby, MATCH 26(1991)111.
[7] J.R. Brisson, *Lipids in Human Nutrition* (MTP Press, Lancaster, UK, 1982).
[8] C.E. Soteros and S.G. Whittington, J. Math. Chem. 5(1990)307.
[9] J.V. Knop, W.R. Müller, K. Szymanski and N. Trinajstić, *Computer Generation of Certain Classes of Molecules* (SKTH/Kemija u industriji, Zagreb, 1985).
[10] J.V. Knop, W.R. Müller, K. Szymanski, S. Nikolić and N. Trinajstić, in: *Computational Graph Theory*, ed. D.H. Rouvray (Nova Science, New York, 1990).
[11] M. Randić, J. Chem. Inf. Comput. Sci. 26(1986)136–148.
[12] M. Randić, Croat. Chem. Acta 59(1986)327–342.
[13] M. Randić, S. Nikolić and N. Trinajstić, J. Mol. Struct. (THEOCHEM) 165(1988)213–228.
[14] N. Trinajstić, private communication (19th April 1989).