Univ. Beograd. Publ. Elektrotehn. Fak.
Ser. Mat. 1 (1990), 49–59.

A BRANCH–AND–BOUND TECHNIQUE FOR COUNTING NON–ISOMORPHIC HEXAGONAL SYSTEMS

Dragan M. Acketa, Ratko Tošić

A new technique for counting non–isomorphic general hexagonal systems with n hexagons is presented. The main feature of the method is that it avoids isomorphism tests. Instead, the hexagonal systems of a given size are freely generated within a certain bounded region of the hexagonal grid. The generated duplications (mutually isomorphic hexagonal systems) are eliminated afterwards by using the tables of the number of those non-isomorphic hexagonal systems, which are symmetrical in some sense.

0. INTRODUCTION

The problem of counting non–isomorphic hexagonal systems having the given number of hexagons is a well–known application of a combinatorial enumeration to the studies of benzenoid molecules in organic chemistry (see, e.g., [**2**], [**4**]).

A standard approach to the exhaustive generation of non–isomorphic combinatorial objects within some class C is "extend and reduce". The "extend" step generates all the possible objects of size $n+1$ within C, that are produced by augmentation of all non–isomorphic objects of size n within C. The "reduce" step eliminates isomorphic copies of size $n+1$, leaving exactly one representative for each isomorphic class. Both steps include expensive isomorphism tests: the local isomorphism tests in the first case and by far more expensive global isomorphism tests over the generated catalogues in the second case.

Another approach, due to Reid [**5**], is based on direct constructions of canonical extensions and canonical representatives of isomorphic classes, without producing mutually isomorphic copies. The canonical constructions are structure–dependent; they must take into account the features of isomorphic transformations over the considered structure. Some versions of this approach were also applied to the enumeration of hexagonal systems ([**6**]) and to related triangular and square systems as well ([**3**], [**10**]).

This paper offers a third approach, when the domain of hexagonal systems is

AMS Subject Classification (1980): 05C30

considered and when the sole enumeration, without producing catalogues, is required. Expensive isomorphism tests are completely avoided. The method is based on the use of the formerly obtained results concerning the symmetric hexagonal systems; more precisely, those hexagonal systems which have non–trivial symmetry groups. It is to say that the generation of non–isomorphic symmetric hexagonal systems is much facilitated, in comparision with the generation of non-isomorphic general hexagonal systems. The reason lies in the fact that the search in symmetric cases can be restricted to a particular part of the hexagonal system, which is being produced. Consequently, the enumeration of symmetric hexagonal systems has been completed (see, e.g., [**2**], [**7**], [**8**], [**11**]) with much larger numbers of hexagons than in the general case.

The main idea of the proposed method is to generate hexagonal systems within a specially shaped region of the hexagonal grid. The generation of all the non–isomorphic hexagonal systems of a required size is guaranteed. Some of them are produced several times, but all these multiplicities are known exactly; they depend on the inherent symmetries of a particular hexagonal system.

1. DEFINITIONS AND NOTATION

The (infinite) *hexagonal grid* is the figure obtained by tiling the whole plane with regular hexagons of the same size. The *hexagonal system* is a part of the hexagonal grid, which contains a finite number of hexagons (this number is a parameter of the system) and which can be surrounded by a closed non–self–intersecting path. Two hexagonal systems are *isomorphic* if there is an isometric transformation (a combination of translations, rotations and axial symmetries), which makes them coincide. There is a bijection between the hexagonal systems and their boundary closed paths, so the counting the non–isomorphic hexagonal systems can be replaced by the counting of the corresponding closed paths.

The hexagonal grid is further represented by the "brick–wall type" coordinate system. The vertices of the hexagons are (in cyclical order) the points with coordinates

$$(x-1, y+1), \quad (x-1, y), \quad (x, y), \quad (x+1, y), \quad (x+1, y+1), \quad (x, y+1),$$

where $x + y$ is odd. In this way, hexagonal grid is embedded in the usual tetragonal grid; it is obtained from the latter grid by excluding the edges of the form $(x, y+1)$—(x, y), where $x + y$ is odd.

The following two features are used to obtain a bounded region of the hexagonal grid of a reasonable size, which would be sufficient to comprise all the non–isomorphic hexagonal systems with a given number of hexagons:

- each hexagonal system S has a left–most hexagon (which has a vertex with the smallest x–coordinate),
- among the left–most hexagons there exists the lowest one (with the smallest y–coordinate).

The unique hexagon constructed in this way is labelled by 1. Its vertices (in cyclical order) will have the coordinates $(0, 0)$, $(1, 0)$, $(2, 0)$, $(2, 1)$, $(1, 1)$, $(0, 1)$ in the associated

coordinate system. This hexagon is the common initial hexagon for all the hexagonal systems which are going to be constructed.

Hexagonal systems having at most n hexagons are counted inside the bounded region of the special form $Cage(n)$. $Cage(5)$ is presented in *Fig. 1*. Note that the hexagons denoted by the number j belong to $Cage(i)$, for each $i \geq j$:

Fig. 1

2. AN ALGORITHM FOR THE RECURSIVE GENERATION OF CLOSED PATHS ASSOCIATED TO HEXAGONAL SYSTEMS

Hexagonal systems can be uniquely represented by their contours, i.e., by closed paths without intersections, which frame the hexagons of the system. On the other hand, a path is constructed by applying an advancing process (to be defined below), which is applied to each point of that path, which is being generated.

A boolean matrix *Liber* plays a central role in directing the path. It determines whether a point is free or not for the continuation of a path.

The initial values of *Liber* are used to determine the search area – $Cage(n)$ by putting the values "`TRUE`" in the interior of $Cage(n)$ and the values "`FALSE`" on its border by the following *PASCAL* procedure "**Initialize**":

```
Procedure Initialize (n : integer;
  var liber: array [−1..2n + 1, −n..n + 1] of boolean);
  begin (* Initialize *)
    (* Liber is primarily initialized to be TRUE
```

```
    within the rectangle which covers the whole cage *)
  for x := −1 to 2n + 1 do
    for y := −n to (n + 1) do liber [x, y] := true;
  (* Upper and lower border of the cage *)
  for x := 0 to n do if (n − x) mod 2 = 0 then begin
    liber [x, n + 1] := false;
    liber [x, −n] := false end;
  (* Left border of the cage – upper part *)
  for y := 2 to n do liber [−1, y] := false;
  (* Left border of the cage – lower part *)
  for y := −n + 1 to 1 do liber [0, y] := false;
  (* Right border of the cage – upper part *)
  for y := 1 to n do liber [2n + 2 − y, y] := false;
  (* Right border of the cage – lower part *)
  for y := 0 downto −n + 1 do liber [2n + 1 + y, y] := false;
end (* Initialize *)
```

Example. The "shell" of true (`T`) and false (`F`) values of $Liber$ in the initial position of $Cage(3)$ seems as follows:

```
      F  F  F  F  F
   F  T  T  T  T  T  F
   F  T  T  T  T  T  T  F
   F  T  T  T  T  T  T  T  F
   F  T  T  T  T  T  T  T  F
      F  T  T  T  T  T  F
      F  T  T  T  T  F
         F  F  F  F
```

The matrix $Liber$ is adjusted during the generation of paths. Its value in a point (x, y) is turned to `FALSE` after the point (x, y) is added to the current path; re–entering (x, y) in such a state would lead to a self–intersection of the path. On the other hand, the `TRUE` value of $liber\,[x, y]$ is returned after the point (x, y) is excluded from the current path; new possibilities for re–entering it are opened in that case.

Example. The effect caused in the matrix $Liber$ after the path has passed through a "free" zone seems as follows:

```
      T  T  T  T  T  T  T  T  T  T
      T  T  T  T  T  T  F  F  F  F  →
      T  T  T  T  F  F  F  T  T  T
⟶     F  F  F  F  F  T  T  T  T  T
      T  T  T  T  T  T  T  T  T  T
```

We proceed with a detailed description of this advancing process:

The common initial point for all the paths that we use is the point $(1, 0)$ – the middle point of the bottom edge of the initial hexagon. In addition, the common final point of all these paths will be $(1, 1)$ (these two points were specially marked on the figure in *Section 1*). Although the initial and the final point do not coincide, all the

constructed paths will be closed, since the path connecting the points $(1,1)$, $(0,1)$, $(0,0)$ and $(1,0)$, in this order, will be always additionally included.

There are six directions, all of which are be shown on an oriented hexagon, both in the usual and in the "brick–wall" representation (see *Fig. 2*).

Fig. 2

Each of these directions may be also viewed as a state associated to an edge before entering a vertex. The initial state is 1. There are two continuations from each vertex (x, y) (the left one and the right one). The direction (state) of these continuations depends on the entering state. Each line of the following transformation table is corresponded to a state (1: through 6:) and contains the coordinates of the next point (denoted as (nx, ny)) and the next state for the left turn (L) and the right turn (R):

$1:$	$L(x+1, y; 2)$	$R(x, y-1; 6)$	$2:$	$L(x, y+1; 3)$	$R(x+1, y; 1)$
$3:$	$L(x-1, y; 4)$	$R(x+1, y; 2)$	$4:$	$L(x-1, y; 5)$	$R(x, y+1; 3)$
$5:$	$L(x, y-1; 6)$	$R(x-1, y; 4)$	$6:$	$L(x+1, y; 1)$	$R(x-1, y; 5)$

The following procedure "**Develop**" regulates the advancing process. Two recursive calls of this procedure are naturally corresponded to the two continuations after each vertex (x, y):

```
Procedure Develop (x, y : integer);
  begin
    liber[x, y] := false;
    length_of_path := length_of_path + 1;
    (* the point (x, y) is added to the current path *)
    if (x = 1) and (y = 1) then
      output a new hexagonal system
    else begin
      Turn left and find nx, ny;
      if liber [nx, ny] then develop (nx, ny);
      Turn right and find nx, ny;
      if liber [nx, ny] then develop (nx, ny);
    end;
    liber [x, y] := true;
    length_of_path := length_of_path − 1;
    (* the point (x, y) is removed from the current path *)
  end; (* Develop *)
```

The main program, which calls the procedure "**Develop**", has the following outlook:

```
begin (* Main *) read(n);
  Initialize (n, liber); Develop;
end.
```

3. COUNTING THE NUMBER OF HEXAGONS

The following lemma offers an easy way for determining the number of hexagons within a hexagonal system, which is surrounded by a traversed path:

Lemma 1. *The number $n(\mathcal{H})$ of hexagons within a hexagonal system $\mathcal{H}$ can be counted on the basis of the following formula:*

$$n(\mathcal{H}) = \frac{1}{2}\left(\sum x(up, \mathcal{H}) - \sum x(down, \mathcal{H})\right),$$

where:

$x(up, \mathcal{H})$ is an x–coordinate with which the step up is done, in the course of traversing the contour of $\mathcal{H}$ (state 3)

$x(down, \mathcal{H})$ is an x–coordinate with which the step down is done, in the course of traversing the contour of $\mathcal{H}$ (state 6)

Proof. The area of a hexagonal system can be partitioned into horizontal sections of width 1 (several of these sections may have the same y–coordinate). The hexagons in such a section are represented by rectangles of size 2×1; their number is exactly one half of the difference of x–coordinates of the vertical borders of the section. ¤

4. THE NUMBER OF NON–ISOMORPHIC HEXAGONAL SYSTEMS

The procedure described in *Section 2* generates only those hexagonal systems which

a) contain the initial hexagon, labelled by 1.

b) are included into $Cage(n)$ for some n.

Lemma 2. *There is not a non–trivial translation which maps one hexagonal system satisfying the properties* a) *and* b) *to another such hexagonal system.*

Proof. If the direction of the translation vector, applied to a hexagonal system satisfying a) and b), belongs to the angle interval $(-90°, +90°]$, respectively to the angle interval $(+90°, -90°]$, then the condition a), respectively the condition b), is not satisfied with the resulting hexagonal system. ¤

The following theorem gives that the multiple appearances of mutually isomorphic copies, which are generated by our algorithm, are exactly known:

Theorem 1. *If a hexagonal system $\mathcal{H}$ has the symmetry group of order k (where $k \in \{1, 2, 3, 4, 6, 12\}$), then there are exactly $12/k$ hexagonal systems isomorphic to $\mathcal{H}$ and satisfying the conditions* a) *and* b).

Proof. A consequence of the previous lemma is that there are at most 12 hexagonal systems in the class $c(\mathcal{H})$, consisting of all those different hexagonal systems satisfying a) and b), which can be obtained by applying isometric transformations to a given hexagonal system $\mathcal{H}$. The hexagonal systems "parallel" to the hexagonal systems in $c(\mathcal{H})$ are obtained by applying to $\mathcal{H}$ the twelve transformations (combinations of rotations and axial symmetries) from the symmetry group D_6 of the hexagonal grid. If the symmetry group of $\mathcal{H}$ is of order k, then there are exactly $12/k$ different hexagonal systems, which can be obtained from $\mathcal{H}$ by applying the symmetries of D_6. ¤

Let $n(G, k)$ denote the number of non–isomorphic hexagonal systems with k hexagons, which have the symmetry group G. Further, let $H(k)$ denote the total number of hexagonal systems with k hexagons, which satisfy the conditions a) and b) (equivalently, which are generated by our algorithm for a given k). Theorem 1 implies the following equation (see also [**9**]):

$$\begin{aligned} H(k) =& n(D_{6h}, k) + 2n(C_{6h}, k) + 2n(D_{3h}, k) + 4n(C_{3h}, k) + \\ &+ 3n(D_{2h}, k) + 6n(C_{2h}, k) + 6n(C_{2v}, k) + 12n(C_s, k). \end{aligned}$$

Remark. The standard denotations of subgroups of D_6 are used here. The denotations D_i and C_i correspond to the dyhedral and cyclic groups of order i respectively. In particular, when the cyclic group C_2 is considered, a distinction is made between the axial symmetry and rotation for 180° (central symmetry).

On the other hand, the partition of non–isomorphic hexagonal systems on k hexagons, with respect to their symmetry groups, gives the following equation:

$$\begin{aligned} NIS(k) =& n(D_{6h}, k) + n(C_{6h}, k) + n(D_{3h}, k) + n(C_{3h}, k) + \\ &+ n(D_{2h}, k) + n(C_{2h}, k) + n(C_{2v}, k) + n(C_s, k), \end{aligned}$$

where $NIS(k)$ denotes the number of non–isomorphic hexagonal systems on k hexagons.

It should be stressed that the counting of non–isomorphic hexagonal systems possessing non–trivial symmetries is much facilitated, for their contours need be only partly generated. Consequently, these numbers are known for much larger values of k than the number $n(C_s, k)$ (which is by far the largest among the eight values $n(G, k)$).

Our method for determining the number $NIS(k)$ is the following: the number $H(k)$ is primarily determined by the algorithm described in *Section 2*. The numbers $n(D_{6h}, k)$, $n(C_{6h}, k)$, $n(D_{3h}, k)$, $n(C_{3h}, k)$, $n(D_{2h}, k)$, $n(C_{2h}, k)$ and $n(C_{2v}, k)$ are taken from the literature. The number $n(C_s, k)$ is determined from the first of the above two equations and substituted in the second one afterwards.

Example. Using $Cage(9)$, 76663 hexagonal systems with 9 hexagons were produced. On the other hand, it is known that there are 1, 5, 7, 36, 178 non–isomorphic hexagonal systems with 9 hexagons possessing the symmetry groups D_{3h}, C_{3h}, D_{2h}, C_{2h}, C_{2v} respectively. Taking into account that each hexagonal system in these classes appears respectively 2, 4, 3, 6, 6 times in the class of the constructed hexagonal systems and that each non–symmetric (= with the trivial symmetry group C_s) hexagonal system appears 12 times, it follows that the number of non–isomorphic non–symmetric hexagonal systems with 9 hexagons is equal to

$$\frac{76663 - 2 \cdot 1 - 4 \cdot 5 - 3 \cdot 7 - 6 \cdot 36 - 6 \cdot 178}{12} = 6278,$$

and finally the number of non–isomorphic hexagonal systems with 9 hexagons is equal to

$$6278 + 1 + 5 + 7 + 36 + 178 = 6505.$$

A table of the calculated numbers of non–isomorphic hexagonal systems seems as follows:

n	factor $H(n)$	1 D_{6h}	2 D_{3h}	4 C_{3h}	3 D_{2h}	6 C_{2h}	6 C_{2v}	12 C_s	$NIS(n)$
1	1	1	0	0	0	0	0	0	1
2	3	0	0	0	1	0	0	0	1
3	11	0	1	0	1	0	1	0	3
4	44	0	1	0	2	1	1	2	7
5	186	0	0	0	2	1	9	10	22
6	813	0	1	1	3	7	12	57	81
7	3640	1	0	1	3	7	39	279	330
8	16590	0	0	0	6	35	61	1333	1435
9	76663	0	1	5	7	36	178	6278	6505

The results obtained in the last column coincide with the figures obtained in [**1**]. The order of running time for $n = 9$ was several hours on PC–AT).

5. SOME COMMENTS ON EFFICIENCY

The main computation problem with the presented algorithm is producing a huge number of hexagonal systems with more than n hexagons within $Cage(n)$. It is useless to count these hexagonal systems, for such a counting is not complete. Let $T(n)$ and $U(n)$ respectively denote the total number of generated hexagonal systems (after the improvements described in 5.1. and 5.2. were implemented) and the number of "useful" hexagonal systems among them. The following table shows the magnitude of this problem:

n	1	2	3	4	5	6	7	8	9
$T(n)$	1	6	36	239	1745	13694	113343	977104	?
$U(n)$	1	4	15	59	245	1058	4698	21208	97951

The data like these imply a high necessity for pruning the search tree, i.e. for reducing the number of recursive calls of the procedure "**Develop**". We sketch three ideas for such improvements:

5.1 The Minimal Length Completion

The maximal perimeter of a hexagonal system having n hexagons is equal to $4n + 2$, [**4**]. This implies that the recursive calls within the call $develop\,(x, y)$ should not be activated (the recursion should be stopped) whenever the following condition is fulfiled:

$$|x - 1| + |y - 1| > (4n + 2) - (3 + \text{length-of-path}),$$

that is, whenever the minimal number of steps needed to complete the contour is greater than the maximal number of the remaining edges with hexagonal systems having n hexagons.

5.2 Forbidden Right Turnings in the Peripheral Area of $Cage(n)$

The right turnings should not be allowed along the peripheral edges of the cage, since there is no possibility in such cases to complete the closed path without self–intersections. This is regulated by the following boolean variable $Allowed$ which is applied in the conjuction with the boolean matrix $Liber$ on the occasion of right turnings:

$$Allowed = \textbf{false} \quad (\text{with } Cage(n)) \quad \text{for:}$$

$$\begin{array}{llllllllllll}
(\ \text{state} = 1 & , & x & - & y & = & 2n & - & 1 & , & y & < 0\ ) \text{ or} \\
(\ \text{state} = 2 & , & x & + & y & = & 2n & & & , & y & > 0\ ) \text{ or} \\
(\ \text{state} = 3 & , & & & y & = & n & & & & &)\ \text{or} \\
(\ \text{state} = 4 \text{ or } 5 & , & x & & & = & 1 & & & , & y & > 0\ ) \text{ or} \\
(\ \text{state} = 4 \text{ or } 5 & , & x & & & = & 2 & & & , & y & < 0\ ) \text{ or} \\
(\ \text{state} = 6 & , & & & y & = & -n & + & 1 & & &)\ ,
\end{array}$$

otherwise $Allowed = \textbf{true}$.

5.3 A Convex Hull Improvement of the Distance

A further improvement of the minimal length completion is reached by observing that the current path cannot be evaded from the left–hand side. This implies that

the minimal number of edges needed to complete the path from a point (x, y) is not smaller than

$$|x - p| + |y - p| + |p - 1| + |q - 1|,$$

where the numbers p and q are respectively for 1 greater than the largest x–coordinate (respectively y–coordinate), which was reached along the path. The above expression should replace the left–hand side of the inequality in 5.1.

In some cases new transition points (besides (p, q)) should be inserted between the current point (x, y) and the final point $(1, 1)$. An elaboration of this idea leads to the necessity to consider the length of the convex hull of the path.

6. CONCLUSION

The proposed method for counting non–isomorphic general hexagonal systems with a given number of hexagons avoids expensive isomorphism tests. It uses the known numbers of non–isomorphic symmetric hexagonal systems and produces all the hexagonal systems containing a fixed hexagon, which might arise within a specially–shaped closed region of the hexagonal grid.

Although the isomorphism tests are avoided, the method is not very efficient due to the counting of a huge number of undesirable hexagonal systems. This failure of the method can be partly overcome by introducing some improvements, which help the "hopeless" recursions to be cut in early stages of the recursion. We hope that some more sophisticated ideas may lead to a considerable improvement of efficiency of the method.

REFERENCES

1. A. T. Balaban, J. Brunvoll, J. Cioslowski, B. N. Cyvin, S. J. Cyvin, I. Gutman, W. C. He, W. J. He, J. V. Knop, M. Kovačević, W. R. Muller, K. Szymanski, R. Tošić, N. Trinajtić: *Enumeration of Benzenoid and Coronoid Hydrocarbons.* Z. Naturfosch., **42a**: 863-870.

2. B. N. Cyvin, J. Brunvoll, S. J. Cyvin: *Enumeration of Benzenoid Systems and other Polyhexes.* preprint.

3. R. Doroslovački, I. Stojmenović, R. Tošić: *Generating and Counting Triangular Systems.* BIT **27** (1987) 18-24.

4. I. Gutman, S. J. Cyvin: *Introduction to the Theory of Benzenoid Hydrocarbons.* Springer–Verlag, 1989.

5. R. C. Reid: *Every one a winner – or how to avoid isomorphism search when cataloguing combinatorial configurations.* Ann. of Discr. Math., **2** (1978) 107-120.

6. I. Stojmenović, R. Tošić, R. Doroslovački: *Generating and Counting Hexagonal Systems*, in: R. Tošić, D. Acketa, V. Petrović [eds]. *Proceedings of the Sixth Yugoslav Seminar on Graph Theory.* Dubrovnik, April 18-19, 1985, University of Novi Sad, Novi Sad, (1986) 189-198.

7. R. Tošić, Z. Budimac, J. Brunvoll, S. J. Cyvin: *Enumeration and Classification*

of Benzenoid Hydrocarbons, Part XII. Catacondensed Systems of Regular Trigonal, J. Mol. Struct. (Theochem), (1990), accepted.

8. R. Tošić, Z. Budimac, J. Brunvoll, S. J. Cyvin: *Enumeration and Classification of Benzenoid Hydrocarbons, Part XIII.* Catacondensed Dihedral Systems J. Mol. Struct. (Theochem), (1990), accepted.

9. R. Tošić, S. J. Cyvin: *An Approach to Enumeration of Benzenoid Hydrocarbons Problem.* submitted.

10. R. Tošić, R. Doroslovački, I. Stojmenović: *Generating and Counting Square Systems*, in: R. Tošić, D. Acketa, V. Petrović, R. Doroslovački [eds]. *Proceedings of the Eighth Yugoslav Seminar on Graph Theory,.* Novi Sad, April 17-18, 1987, University of Novi Sad, Novi Sad, (1988) 127-136.

11. R. Tošić, M. Kovačević: *Generating and Counting Unbranched Catacondensed Benzenoids.* J. Chem. Inf. Comput. Sci., **28, 29** (1988) 29-31.

Institute of Mathematics,
dr. Ilije Djuričića 4, 21000 Novi Sad,
Yugoslavia

(Received October 10, 1990)