

A Constructive Enumeration of Fusenes and Benzenoids

Gunnar Brinkmann
Fakultät für Mathematik
Universität Bielefeld
D 33501 Bielefeld, Germany
gunnar@mathematik.uni-bielefeld.de

Gilles Caporossi
Département de Méthodes Quantitatives en Gestion
École des Hautes Études Commerciales
Montréal, Canada
gilles.caporossi@gerad.ca

Pierre Hansen
GERAD and Département de Méthodes Quantitatives en Gestion
École des Hautes Études Commerciales
Montréal, Canada
pierre.hansen@gerad.ca

Proposed running head: A Constructive Enumeration of Fusenes and Benzenoids

Proofs shall be sent to:

Gunnar Brinkmann
Fakultät für Mathematik
Universität Bielefeld
D 33501 Bielefeld

Abstract

In this paper, a fast and complete method to constructively enumerate fusenes and benzenoids is given. It is fast enough to construct several million non isomorphic structures per second. The central idea is to represent fusenes as labelled inner duals and generate them in a two step approach using the canonical construction path method and the homomorphism principle.

Introduction

A *fusene* is a simple planar 2-connected graph embedded in the plane with all the vertices of degree 2 or 3, all bounded faces hexagons and all vertices not in the boundary of the outer face of degree 3. In nature fusenes occur as planar polycyclic hydrocarbons. The vertices are carbon atoms and every vertex of degree 2 is assumed to have an additional bond with a hydrogen atom. Since the hexagons in planar polycyclic hydrocarbons are not only *combinatorial* hexagons, but geometrically close to regular hexagons, especially those fusenes occur very often that can be build of regular hexagons in the plane without overlap, or combinatorially speaking: that are a subgraph of the regular hexagonal lattice. Fusenes with this property are called *benzenoids*. For an introduction into the mathematical and chemical properties of these structures see [11] or [15].

Due to their importance there is a long history of algorithms for enumerating benzenoids and fusenes, see e.g. [1] [2] [3] [6] [7] [8] [9] [10] [17] [18] [21] [22] [23] [24] [25]. All the algorithms proposed have been of a constructive nature, that is: they do not only determine the number of structures, but they are also able to output one element of every isomorphism class (except for [25] where isomorphic structures are constructed and only the numbers of non-isomorphic ones are determined). This is an important property in order to be able to examine the structures for certain mathematical or chemical properties. Nevertheless also the numbers of benzenoids are interesting. In this direction recent work by Vöge, Guttmann and Jensen (see [26]) may well be regarded as a breakthrough. They developed a nonconstructive method to enumerate benzenoids and gave the number of structures with up to 35 hexagons – a number close to $6 \cdot 10^{21}$ that is impossible to reach by constructive enumeration, where it is required that every structure is really formed in the memory and possibly even outputted. For fusenes no nonconstructive method is known so far.

Though maybe larger and larger numbers of benzenoids can be determined in the future, we have good reason to believe that our algorithm is the endpoint of the long series of algorithms for **constructive** enumeration of benzenoids and fusenes: Since now algorithms for nonconstructive enumeration exist, constructive enumeration is only interesting in order to really examine the structures for their chemical and mathematical properties. This means that any structure has to be coded and maybe outputted. The algorithm described in this paper is so fast, that coding and outputting the structures is in fact the part dominating the running time. Using e.g. an adjacency list as the code for the fusenes, already for 15 hexagons a C-program just reading the data from a pipe (to be exact: just consisting of the line `while ((c=getc(stdin)) != EOF);`) takes already about 85 percent of the time of the generation program constructing, coding and writing the structures and more than 220 times as long as needed to form the structures to be outputted in memory. The relative amount of time needed for reading and writing even increases with the number of hexagons. So assuming the same codes to be outputted, there is simply not much room for improvement. Of course a large amount of CPU can be saved by including the code to test the structures into the generation code and use the same memory and datastructures as the generator. Nevertheless our argument shows that most likely any nontrivial test would be far more expensive than the generation.

For benzenoids the generation rate, that is the number of nonisomorphic structures listed per second, is

smaller than for fusenes if the graphs are just constructed in memory, but since outputting the graphs is by far the most time consuming part, the generation rates almost agree if e.g. adjacency lists have to be outputted.

Graphs are denoted by $G = (V, E)$ with V the set of vertices and E the set of edges. The degree of a vertex $v \in V$ is denoted by $\deg(v)$. Though multigraphs do occur in this paper, all the cases that have to be discussed in detail deal with simple graphs, so we adapted our notation to them.

All graphs are assumed to be connected and all embedded graphs to be embedded in the plane, described by a rotation system like e.g. in [13] or [20]. So we can represent every edge by two directed edges inverse to each other (we write e^{-1} for the inverse of edge e) and every vertex v comes with a *cyclic ordering* of the edges starting at v , which we interpret as clockwise. An isomorphism of an embedded graph is a graph theoretic isomorphism either preserving all cyclic orderings or reversing them all. The *boundary cycle* of a face f is a cyclic sequence of vertices and directed edges obtained by starting at a directed edge e_0 with f on the left and having arrived at edge e listing the edge and its start vertex and proceeding with the edge e' that comes after e^{-1} in the cyclic ordering of the endpoint of e . The boundary cycle is finished when we are back at e_0 . Note that the boundary cycle is not necessarily a simple cycle. A proper subsequence of the boundary cycle is called a boundary path.

The algorithm

Our algorithm to list benzenoids with a given number n of hexagons first constructs all fusenes with n hexagons and filters them for structures where the boundary cycle is a simple curve in the hexagonal lattice (to be exact: our test uses the dual triangular lattice – the reason for this will soon become clear). It is obvious that – assuming one has built a copy of a large enough part of the lattice in the memory of the computer (which has to be done only once for all the structures) – this can easily be done in running time linear in the length of the boundary (even with a very small coefficient for the linear term), so also linear in n in the worst case. We also tried to use information from the filtering in order to allow a look ahead and avoid the construction of fusenes that do not correspond to benzenoids, but the amount of time needed for this look ahead turned out to be larger than the amount of time saved by not constructing and testing some fusenes. In practice straightforward filtering turned out to be fastest, though this might change for larger numbers of hexagons. What remains is to describe our algorithm to list fusenes:

The *inner dual* of a graph embedded in the plane is its dual graph with the vertex corresponding to the unbounded region removed. So it is a subgraph of the dual graph. While there is a one-to-one correspondence between embedded graphs and their duals, this is not the case for inner duals as can be seen in figure 1.

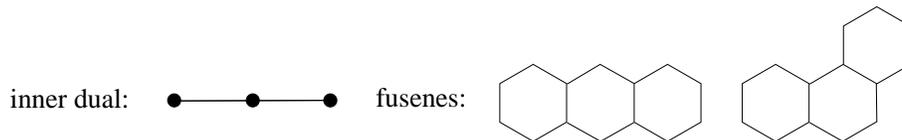


Figure 1: Two fusenes with the same inner dual.

The inner duals of fusenes (which will be called *id-fusenes* in the remainder of this paper) are simple graphs. This can be shown directly, but is also a consequence of the results about boundary lengths in [12] and [16] for the general case.

The following lemma gives a characterization of id-fusenes.

Lemma 1 *A planar embedded graph is the inner dual of a fusene with n hexagons, if and only if*

- (a) *it has n vertices;*
- (b) *it is connected;*
- (c) *all bounded faces are triangles;*
- (d) *vertices not in the boundary have degree 6;*
- (e) *for all vertices the degree plus the number of times it occurs in the boundary cycle of the outer face is at most 6.*

A proof of the lemma by induction on n is very easy and left to the reader. In fact a recursive structure for the class of embedded graphs defined by properties (a) to (e) in Lemma 1 is given later and can be used for the proof.

A labelled embedded graph is an embedded graph together with a map from the set of all *angles*, that is pairs e, e' with e' following e in the cyclic order of their common start vertex, into the set \mathbb{N}_0 . Two labelled embedded graphs are isomorphic if there is an isomorphism of the embedded graph with the property that all angles are mapped onto angles with the same labels. The labelled inner dual of a planar graph is obtained by labelling every angle with the number of edges in the cyclic order of the dual graph between e and e' . Since the edges counted are in the dual, but not the inner dual, their endpoint is the vertex corresponding to the unbounded face. If the graph has no bridges, so that the dual has no loops (especially no loops at the vertex corresponding to the outer face), we can reconstruct the dual of the planar graph from its labelled inner dual and the graph from its dual. So there is a one-to-one correspondence between bridgeless planar graphs and their labelled inner duals and two such graphs are isomorphic if and only if their labelled inner duals are.

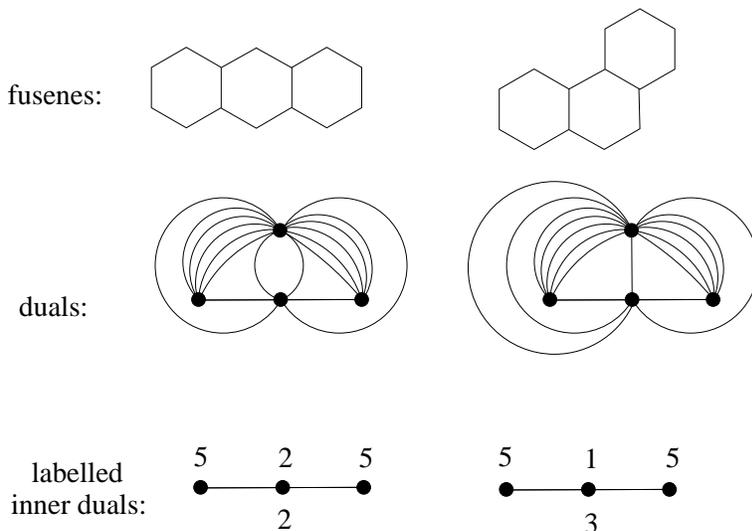


Figure 2: Two fusenes, their duals and labelled inner duals.

Remark 1 *A labelled embedded graph G is the labelled inner dual of a fusene, if and only if*

- *G is an id-fusene;*
- *all angles e, e' with the unbounded face between e and e' have a nonzero label while all angles e, e' with a bounded face between e and e' are labelled zero;*
- *for every vertex v the sum of its degree and the labels of all angles centered at v equals 6.*

The proof is again straightforward.

We will construct fusenes as their labelled inner duals. This will be done in a two step approach. For given number n of hexagons we will first construct all id-fusenes with n vertices $1, \dots, n$. Isomorphism rejection will be done by McKay's canonical construction path method (see [19]). Then we will label them in every possible way. We will make sure not to construct isomorphic labelled inner duals – or equivalently isomorphic fusenes – by the labelling using the *homomorphism principle* developed by A. Kerber and R. Laue (see e.g. [14]). In [4] a survey on various isomorphism rejection methods, including these two, is given.

Constructing all non isomorphic inner duals

A boundary segment of an id-fusene is an edge e of the boundary cycle together with an integer number $l > 0$. The vertices of the boundary segment are the endvertices of e and the next $l - 1$ edges of the boundary cycle in this order.

In order to simplify notation, in the case where the id-fusene is a single isolated vertex, this vertex is defined to be the vertex of a special boundary segment without an edge and with $l = 1$.

A boundary segment is defined to be augmenting if

- $l \leq 3$;
- the first and last vertex have degree at most 4;
- if $l = 1$ the only vertex in the segment has degree at most 3;
- if $l = 3$ and the middle vertex occurs only once in the boundary, it has degree 5.

In order to construct all id-fusenes we use the following lemma:

Lemma 2 *All id-fusenes can be constructed from the inner dual of a single hexagon (that is the single vertex 1) by successively adding vertices and connecting them to each vertex of an augmenting boundary segment.*

(To be exact: the cyclic order of the edges at the new vertex is reverse to the order of their endvertices in the segment. At the vertices of the boundary segment the new edges are inserted into the cyclic order so that for the k -th vertex the next edge of the new edge is the k -th edge following the defining edge of the segment in the boundary cycle. In case of a single vertex in the id-fusene there is just one way to add edges.)

Proof: We will prove that the class of graphs with properties (a) to (e) of lemma 1 can be constructed this way. The recursive structure can then be used for an easy proof that this class is exactly the class of id-fusenes. We will use the term id-fusene for graphs in that class already in this proof.

We will show that every id-fusene – except the single vertex – has an ancestor from which it can be constructed. To this end we show that it can be reduced by the inverse of the operation given in the lemma.

Let $G = (V, E)$ be an id-fusene with at least 2 vertices, bl the length of the boundary cycle, $\text{def}(v) := 6 - \deg(v)$ the deficit of a vertex v and $\text{def} := \sum_{v \in V} \text{def}(v)$. Then some simple calculations involving the euler formula give $\text{def} = 6 + 2 \cdot \text{bl}$.

For $i \in \{1, 2, 3\}$ let V_i denote the set of vertices that occur i times in the boundary cycle and note that the degree of a vertex is at least as large as the number of times it occurs in the boundary cycle.

So we have $\text{bl} = |V_1| + 2 \cdot |V_2| + 3 \cdot |V_3|$ and

$$\sum_{v \in V_1} \text{def}(v) + \sum_{v \in V_2} \text{def}(v) + \sum_{v \in V_3} \text{def}(v) = 6 + 2 \cdot |V_1| + 4 \cdot |V_2| + 6 \cdot |V_3| \text{ implying}$$

$$\sum_{v \in V_1} \text{def}(v) = 6 + 2 \cdot |V_1| + (4 \cdot |V_2| - \sum_{v \in V_2} \text{def}(v)) + (6 \cdot |V_3| - \sum_{v \in V_3} \text{def}(v))$$

$$\sum_{v \in V_1} \text{def}(v) \geq 6 + 2 \cdot |V_1|$$

So there are vertices that occur only once in the boundary cycle and have a deficit of at least 3 and therefore a degree of at most 3. These vertices can be removed without disconnecting the id-fusene with the result a smaller id-fusene where the (former) neighbourhood of these vertices is an augmenting boundary segment so that extension gives the original id-fusene. ■

In order to apply McKay's canonical construction path method, we need a *canonical choice function*, that is a function f from the set of all id-fusenes on the vertices $1, \dots, n$ to the set $2^{\{1, \dots, n\}}$ of subsets of $\{1, \dots, n\}$, such that, given an id-fusene G , f chooses an orbit of vertices under the automorphism group of G . Precisely:

- (a) For each id-fusene G , $f(G)$ is a vertex orbit of the automorphism group of G consisting of boundary vertices that occur only once in the boundary cycle and have a degree of at most 3.
- (b) For each pair of id-fusenes G, G' , any isomorphism $G \rightarrow G'$ maps $f(G)$ onto $f(G')$.

For a given canonical choice function our algorithm to construct all id-fusenes with n vertices starts with the id-fusene with a single vertex and extends the structures with $n' < n$ vertices recursively in the following way:

Step 1: Compute the orbits of augmenting boundary segments.

Step 2: Connect the new vertex $n' + 1$ with one representative of each orbit in turn.

Step 3: Compute the canonical choice function for the extended id-fusene and accept it if and only if the vertex $n' + 1$ is in the canonical orbit.

In step 1 we have to define the image of an augmenting boundary segment under an automorphism. We define the action of an element γ of the automorphism group of the id-fusene on an augmenting boundary segment (e, l) to be (e', l) with $e' = \gamma e$ in case γ is orientation preserving and e' the edge l edges before the inverse of γe in the boundary cycle in case γ is orientation reversing.

Theorem 3 *The algorithm described above accepts exactly one member of every isomorphism class of id-fusenes on n vertices.*

Proof: First we will show that every id-fusene is accepted at least once. This is clearly true for the id-fusene with just one vertex, so assume it is true up to $n - 1$ vertices and let G be an id-fusene with n vertices. If v is a vertex in the canonical orbit of G , e_0 the (unique) boundary edge with endpoint v and S the set of neighbours of v , then $G' = G - \{v\}$ is a smaller id-fusene and by induction an isomorphic copy of it has been accepted. In order to simplify notation, w.l.o.g. assume that G' itself was accepted. Then (e, l) with e the edge before e_0 in the boundary cycle of G and $l = |S|$ is an augmenting boundary segment so that S is the set of its vertices and using it for augmentation we would obtain G . One augmenting boundary segment (e', l) in the orbit of (e, l) was extended to form a graph \bar{G} by adding the new vertex n and connecting it to all vertices in (e', l) . In this case the automorphism mapping (e, l) onto (e', l) extended by mapping v to n is an isomorphism between G and \bar{G} . Since v is in the canonical orbit of G , n is in the canonical orbit of \bar{G} , so \bar{G} is accepted.

Now assume that two different isomorphic id-fusenes G, G' are accepted. So the vertex n is in the canonical orbits of G and G' , implying that there is an isomorphism $\gamma : G \rightarrow G'$ mapping n to n , so $\gamma|_{\{1, \dots, n-1\}} : G - \{n\} \rightarrow G' - \{n\}$ is an automorphism of $G - \{n\} = G' - \{n\}$ which are identical by induction, mapping the boundary segment corresponding to n onto the one corresponding to n' . This is a contradiction since only one element of every orbit is used for extension. ■

For the canonical choice function $f()$ we applied, we used the code associated to a directed edge and an orientation (clockwise, counterclockwise) that is constructed as follows:

Label the starting vertex 1, and the endvertex 2. Then label the endvertices of the remaining edges starting at the same vertex 3, resp 3 and 4 in the order given by the orientation. The inverse of the edge along which a vertex was seen when it was labelled is called its reference edge. Having labelled all the vertices around a vertex labelled i , we start at the reference edge of the vertex labelled $i + 1$ (which is well defined at that time due to the graph being connected) and inspect each neighbouring endvertex in the given orientation. If an endvertex is not yet labelled, it is assigned the smallest number not used so far.

This process is repeated until all the vertices have been labelled. The code corresponding to the edge and orientation is then generated by concatenating the list of neighbours for each vertex starting with the endpoint of the reference edge, and separating each list by a zero.

Remark 2 *The code just described can be computed in linear time and there is an orientation preserving automorphism of the graph mapping e onto e' if and only if the corresponding codes are identical for the same orientation and an orientation reversing automorphism if and only if the codes are identical for different orientations.*

We defined the image of our canonical choice function as follows: First we compute the set S_0 of all vertices that occur only once in the boundary cycle and choose the subset S_1 of S_0 of vertices with minimal degree. Among all vertices in S_1 we choose the subset S_2 of those that have a neighbour of maximal degree. For all vertices v in S_2 we compute all codes corresponding to edges starting at v and ending at a vertex of maximal degree. The set of canonical vertices is the set of all those vertices in S_2 that are the starting point of an edge with smallest code.

Due to the remark above, this is an orbit of vertices and in fact we also get the automorphism group of the graph which – in case the graph is accepted – is needed in step 1 for the next iteration as well as for computing nonequivalent labellings of the inner dual.

Since in the worst case the code has to be constructed for a number of edges that is linear in the number of vertices of the id-fusene, we get a total running time of the canonicity checking routine that is quadratic in the number of vertices.

From inner duals to labelled inner duals

By definition an isomorphism between two labelled inner duals is an isomorphism of the unlabelled inner dual too. So labelling the inner duals in every possible way, isomorphic labelled id-fusenes come as labellings of the same inner dual. Therefore the isomorphism of the unlabelled inner duals must be an automorphism.

We fix an arbitrary edge e_0 in the boundary cycle of each id-fusene and let E_0 denote the set of all pairs (e, o) with e in the orbit of e_0 and $o = 0$ in case e is the image of e_0 under an orientation preserving automorphism and $o = 1$ else. Note that for some edges e both, $(e, 0)$ and $(e, 1)$ can be in E_0 .

For all angles that are not on the boundary, the labels are 0, so it is enough to record the labels for angles on the boundary. For a pair $(e, 0)$ we define the corresponding code to be the cyclic sequence of labels of the angles along the boundary cycle in the order in which they occur when starting at e . Of course only those entries have to be listed that correspond to angles where the label is not determined uniquely by the structure, e.g. since the vertex occurs only once in the boundary cycle. For $(e, 1)$ we define the code as the one obtained in the mirror image (or equivalently: the one obtained when traversing the boundary cycle in opposite direction).

A labelled id-fusene is accepted if and only if there is no pair in E_0 corresponding to a smaller code than $(e_0, 0)$.

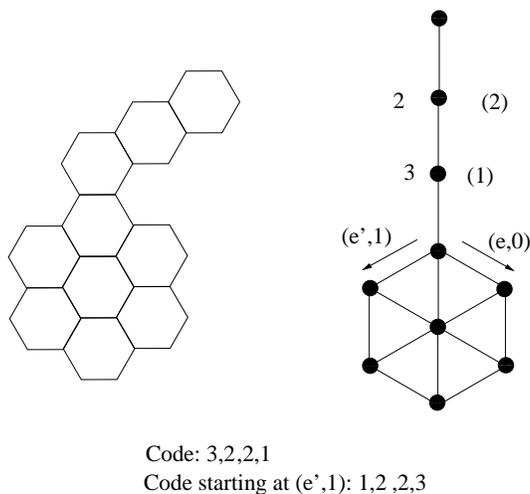


Figure 3: Two equivalent starting edges and the corresponding codes.

The time needed by this test is proportional to the length of the boundary times $|E_0|$, so it is quadratic in the worst case. But in by far most of the cases this time is constant: If the id-fusene has a trivial group, $(e_0, 0)$ is the only element of E_0 , so there is no need for computing any code and the labelled id-fusene can be accepted without any tests. This is e.g. the case for 99.9994 % of the id-fusenes with 26 vertices and the main reason for the high generation rate of the program.

Theorem 4 *Exactly one element of every isomorphism class of labelled id-fusenes is accepted.*

Proof: Since all possible labellings are constructed, the one with minimal code is also constructed and of course accepted, so all there is to show is that no two isomorphic labelled id-fusenes are constructed. Two such fusenes F_0, F_1 of course have different codes corresponding to $(e_0, 0)$. W.l.o.g. F_0 has the smaller one. The isomorphism mapping F_0 onto F_1 must map $(e_0, 0)$ onto some edge (e_1, l) in F_1 , but then for F_1 the code corresponding to (e_1, l) is the same as that for $(e_0, 0)$ and F_0 – so F_1 would not have been accepted. ■

Results

The source code of the C-program based on this algorithm can be obtained from any of the authors. It computes also some statistics about the structures generated (e.g. the numbers of structures grouped with respect to the automorphism group, the corresponding chemical formula or the combination of both), but since this data is mainly interesting for chemists, it will be published in a chemical journal. The total numbers of structures were already announced in [5]

hexagons	fusenes	benzenoids	inner duals	inner duals with trivial group	app. CPU time fusenes (sec)
1	1	1	1	1	0
2	1	1	1	0	0
3	3	3	2	0	0
4	7	7	4	0	0
5	22	22	8	0	0
6	82	81	21	5	0
7	339	331	53	22	0
8	1 505	1 435	151	90	0
9	7 036	6 505	458	342	0
10	33 836	30 086	1 477	1 247	0
11	166 246	141 229	4 918	4 491	0.3
12	829 987	669 584	16 956	16 095	1.2
13	4 197 273	3 198 256	59 494	57 906	5.3
14	21 456 444	15 367 577	212 364	209 170	19
15	110 716 585	74 207 910	766 753	760 830	95
16	576 027 737	359 863 778	2 796 876	2 784 913	425
17	3 018 986 040	1 751 594 643	10 284 793	10 262 649	1 952
18	15 927 330 105	8 553 649 747	38 096 072	38 051 063	9 615
19	84 530 870 455	41 892 642 772	141 998 218	141 914 613	46 663
20	451 069 339 063	205 714 411 986	532 301 941	532 131 882	225 100
21	2 418 927 725 532	1 012 565 172 403	2 005 638 293	2 005 320 952	1 205 600
22	13 030 938 290 472	4 994 807 695 197	7 592 441 954	7 591 794 561	
23	70 492 771 581 350	24 687 124 900 540	28 865 031 086	28 863 820 538	
24	382 816 374 644 336	122 238 208 783 203	110 174 528 925	110 172 051 829	
25	2 086 362 209 298 079		422 064 799 013	422 060 152 511	
26	11 408 580 755 666 756		1 622 379 252 093	1 622 369 728 951	

The largest case we ran on a single machine was for 21 hexagons, so the times are only given up to that number of hexagons. They are normalized to a 133 Mhz Pentium PC with Linux operating system. For 21 hexagons the computer listed about 2 Million fusenes per second in average. For the larger cases we distributed the jobs on a large amount of **very** different machines, most of them 133 Mhz Pentium PCs, but also some old Sun SLC and DEC alphas. For these cases we did not really form all the structures in the memory, but just computed their number in case of trivial symmetry of the underlying id-fusene. So

in fact we did not really constructively enumerate these structures, since this time they were not needed to run any tests on them. Of course they could have been formed in memory with a very small amount of extra effort per structure.

For 26 hexagons the total amount of CPU used was almost 8 years and the generation rate varied from 973 829 fusenes per second to 1 538 082 050 fusenes per second depending on the case, giving an average of 46 000 000 fusenes per second.

In the case of benzenoids with 24 hexagons the total amount of CPU on the same cluster of machines was almost 13 years with an average generation rate of 300 000 benzenoids per second. The largest case run on a single machine was the case of 19 hexagons run on a 133 Mhz Pentium PC. It took about 32 hours and had an average generation rate of 320 000 benzenoids per second.

References

- [1] A.T. Balaban, J. Brunvoll, J. Cioslowski, B.N. Cyvin, S.J. Cyvin, I. Gutman, W.J. He, J.V. Knop, M. Kovačević, W.R. Müller, K. Szymanski, R. Tošić, and N. Trinajstić. Enumeration of benzenoid and coronoid hydrocarbons. *Z. Naturforsch.*, 42a:863–870, 1987.
- [2] A.T. Balaban and F. Harary. Enumeration and proposed nomenclature of benzenoid cata-condensed polycyclic aromatic hydrocarbons. *Tetrahedron*, 24:2505–2516, 1967.
- [3] K. Balasubramanian, J.J. Kaufmann, W.S. Koski, and A.T. Balaban. Graph theoretical characterization and computer generation of certain carcinogenic benzenoid hydrocarbons and identifications of bay regions. *J. Comput. Chem.*, 1:149–157, 1983.
- [4] G. Brinkmann. Isomorphism rejection in structure generation programs. In P. Hansen, P.W. Fowler, and M. Zheng, editors, *Discrete Mathematical Chemistry*, volume 51 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 25–38, 2000.
- [5] G. Brinkmann, G. Caporossi, and P. Hansen. Numbers of benzenoids and fusenes. *match*, 43:133–134, 2001.
- [6] J. Brunvoll, B.N. Cyvin, and S.J. Cyvin. Enumeration of benzenoid systems and other polyhexes. *Top. in Curr Chem.*, 162:65–180, 1992.
- [7] J. Brunvoll, S.J. Cyvin, B.N. Cyvin, Z. Fuji, and X.F. Guo. Theory of helicenic hydrocarbons 4. further enumeration. *Structural Chemistry*, 7:119–130, 1996.
- [8] G. Caporossi and P. Hansen. Enumeration of polyhex hydrocarbons to $h=21$. *J. Chem. Inf. Comput. Sci.*, 38:610–619, 1998.
- [9] G. Caporossi, P. Hansen, and M. Zheng. Enumeration of fusenes to $h = 20$. In *Discrete Mathematical Chemistry DIMACS Workshop March 23-25 1998*, pages 63–78, 2000.
- [10] B.N. Cyvin, S.J. Xiaofeng, S.J. Cyvin, and F. Zhang. Enumeration of helicenes. *Chemical Physics Letters*, 188:537–542, 1992.
- [11] J.R. Dias. *handbook of polycyclic hydrocarbons*. elsevier, 1987. Part A: benzenoid hydrocarbons.
- [12] J. Greinus. Patches mit minimaler Randlänge. Diplomarbeit, Bielefeld 2001.
- [13] J.L. Gross and T.W. Tucker. *Topological Graph Theory*. John Wiley and Sons, 1987.

- [14] R. Grund, A. Kerber, and R. Laue. MOLGEN – ein Computeralgebrasystem für die Konstruktion molekularer Graphen. *match*, 27:87–131, 1992.
- [15] I. Gutman and S.J. Cyvin. *Introduction to the theory of benzenoid hydrocarbons*. Springer Verlag, 1989.
- [16] F. Harary and H. Harborth. Extremal animals. *J. of Comb., Inf. & Syst. Sci.*, 1(1):1–8, 1976.
- [17] W.J. He, W.C. He, Q.X. Quang, J. Brunvoll, and S.J. Cyvin. Supplement to enumeration of benzenoid and coronoid hydrocarbons. *Z. Naturforsch.*, 43a:693–694, 1988.
- [18] J.V. Knop, W.R. Müller, K. Szymanski, and N. Trinajstić. Use of small computers for large computations: Enumeration of polyhex hydrocarbons. *J. Chem. Inf. Comput. Sci.*, 30:159–160, 1990.
- [19] B.D. McKay. Isomorph-free exhaustive generation. *Journal of Algorithms*, 26:306–324, 1998.
- [20] B. Mohar and C. Thomassen. *Graphs on Surfaces*. Johns Hopkins University Press, 2001.
- [21] W.R. Müller, K. Szymanski, and J.V. Knop. On counting polyhex hydrocarbons. *Croat. Chem. Acta*, 62:481–483, 1989.
- [22] W.R. Müller, K. Szymanski, J.V. Knop, S. Nikolić, and N. Trinajstić. On the enumeration and generation of polyhex hydrocarbons. *J. Comput. Chem.*, 11:223–235, 1990.
- [23] S. Nikolić, N. Trinajstić, J.V. Knop, W.R. Müller, and K. Szymanski. On the concept of the weighted spanning tree of dualist. *J. Math. Chem.*, 4:357–375, 1990.
- [24] I. Stojmenović, R. Tošić, and R. Doroslovacki. Generating and counting hexagonal systems. In *Graph Theory, Proceedings of the sixth Yugoslav Seminar on Graph Theory*. 1985.
- [25] R. Tošić, D. Mašulović, I. Stojmenović, J. Brunvoll, S.J. Cyvin, and B.J. Cyvin. Enumeration of polyhex hydrocarbons to $h=17$. *J. Chem. Inf. Comput. Sci.*, 35:181–187, 1995.
- [26] M. Vöge, A.J. Guttmann, and I. Jensen. On the number of benzenoid hydrocarbons. submitted.