MATCH
Communications in Mathematical and in Computer Chemistry

MATCH Commun. Math. Comput. Chem. **63** (2010) 533-552

ISSN 0340 - 6253

CaGe – a Virtual Environment for Studying Some Special Classes of Plane Graphs – an Update

Gunnar Brinkmann*
Applied Mathematics
and Computer Science
Krijgslaan 281 - S9
Ghent University
B 9000 Ghent, Belgium

Olaf Delgado Friedrichs†
ANU Supercomputer Facility
Australian National University
Canberra ACT 0200, Australia

Sebastian Lisken ‡
Teichstr. 32
D 33615 Bielefeld

Adriaan Peeters§
Applied Mathematics
and Computer Science
Krijgslaan 281 - S9
Ghent University
B 9000 Ghent, Belgium

Nicolas Van Cleemput¶
Applied Mathematics
and Computer Science
Krijgslaan 281 - S9
Ghent University
B 9000 Ghent, Belgium

(Received July 18, 2009)

Abstract

CaGe is an environment for generating and visualizing certain specialized classes of plane graphs. In this article we describe the classes of graphs you can generate with *CaGe* and the options offered by the user interface.

*Gunnar.Brinkmann@ugent.be
†olaf.delgado@googlemail.com
‡lisken@mathematik.uni-bielefeld.de
§apeeters@lashout.net
¶Nicolas.VanCleemput@ugent.be

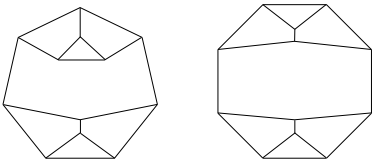

Figure 1: Two cubic graphs that are isomorphic as graphs but not as plane graphs.

Introduction

Most of the graph classes that can be generated by *CaGe* contain *plane* graphs that are also relevant in chemistry. The word *plane* is used here in the mathematical sense which means that the graph is equipped with a cyclic order of the edges around a vertex that defines a combinatorial embedding without crossing edges in the plane – or equivalently on the surface of a sphere. So benzenoids are plane graphs as had to be expected, but fullerenes are also plane graphs, though the molecules are not *geometrically flat*.

In case of graphs that are not 3-connected the concept of *isomorphic* **plane** *graphs* and *isomorphic graphs* differ. While isomorphism between graphs only requires a bijection between the vertex sets that respects the edges, isomorphism of plane graphs also requires that the cyclic order of the edges around the vertices is respected. Either all orders must be identical or all orders must be exactly reversed. Figure 1 gives an example of two graphs that are isomorphic as graphs but not as plane graphs. Whitney's theorem implies that for 3-connected graphs these concepts agree. When embedded in 3-space, there is not necessarily a recognizable order around the vertices. So if two nonisomorphic plane graphs are displayed that are not 3-connected, they can be isomorphic as graphs and it is possible that the drawings cannot be distinguished.

A first version of *CaGe* was described in [1] and since then *CaGe* was used by various researchers for much varying purposes, see [2],[3],[4] for example applications. The first version of *CaGe* was based on Tcl/Tk, but due to several problems with the speed and the development of Tcl/Tk, we decided that instead of repeatedly updating *CaGe*, we should better develop and support a new version based on Java. This Java based version was first released in 2001 and continuously updated.

The version described here is a descendant of that version. The changes applied to

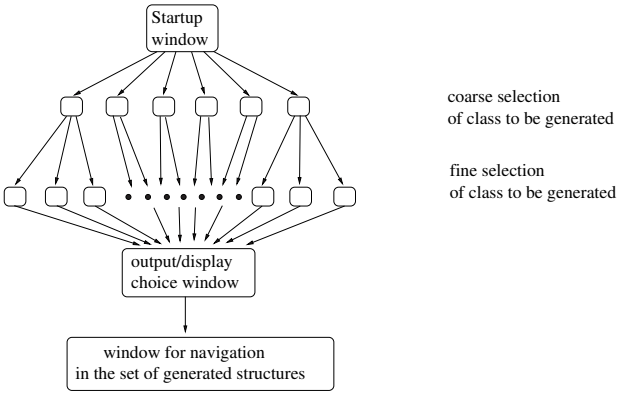

Figure 2: The navigation inside *CaGe*.

that initial java version are described on [5]. Compared to the version described in [1] it offers various new features and generators for more classes of graphs. Furthermore it is more portable now and runs under unix/GNU-Linux and Mac OSX.

How to use *CaGe*

The principal structure of *CaGe* is shown in Figure 2. First one chooses the class of structures to be generated. This is done in 2 steps – first a coarse choice and then a refinement step. The window in Figure 3 shows the possible first, coarse choices of the class of graphs to be generated. Clicking the button with the class of ones choice, a new window appears where one can refine the class and give some parameters. Figure 4 shows the window for triangulations. The possible classes that can be chosen will be explained in the next part.

In some cases a certain class of graphs can be generated in more than one way. The windows reflect the possibilities of the underlying generators and sometimes two generators that have different possible options allow to combine options in a way that as a result the same classes are generated by both.

Having chosen the refinements one clicks *next* and the window for the output options appears (see Figure 5). Choosing the appropriate options and clicking *next* again, the generation starts and in case the options for 2-dimensional and/or 3-dimensional display

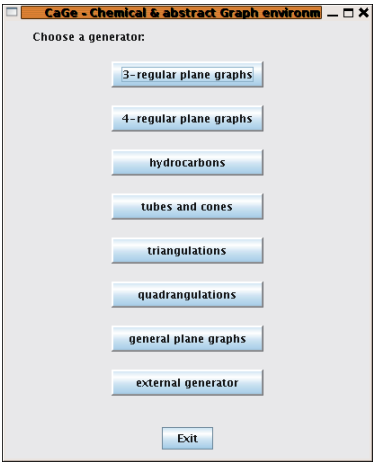

Figure 3: The starting window of *CaGe*.

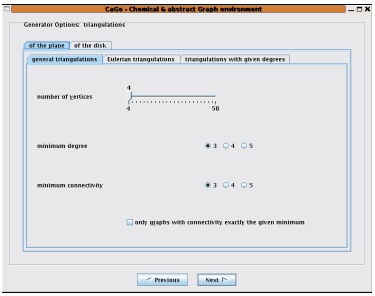

Figure 4: The window showing the refined classes of triangulations that can be chosen to be generated.

are chosen the display windows (see Figure 7 and Figure 8) appear. Furthermore the navigation window (see Figure 6) appears that allows to choose which graph is displayed. One can move forward ("advance") by a single graph or by a larger step. The default step size is 10. One can also let the stream of generated graphs flow by until one stops the flow. Moving to a previous graph is only possible for graphs that one has previously stopped at; graphs that have been skipped are lost.

Classes of graphs

The following classes of graphs can be generated:

3-regular plane graphs

- **Fullerenes**

  Fullerenes are 3-regular plane graphs with all faces of size 5 or 6. The default is to generate all fullerenes with a given number of vertices that has to be chosen. The generation can be restricted in the following ways:

  – IPR fullerenes – that is fullerenes where no two pentagons share an edge.

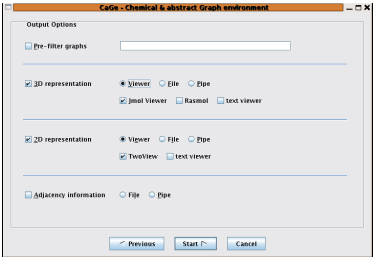

Figure 5: The window where the form of the output is chosen.

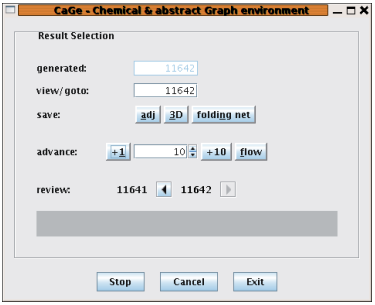

Figure 6: The window with the options that allow navigation in the set of generated structures.

– Fullerenes with a given symmetry group. This option is implemented as a filter, so in case of groups that do not occur very often (or not at all . . .), the generation will be very inefficient.

The user can also choose to output (for display or further tests) the duals of the fullerenes (that is: triangulations with vertices of degree 5 and 6) instead of the fullerenes. Because the number of vertices of the fullerene is given, in the dual the number of faces is fixed.

The algorithm implemented in this generator is described in [6].

- **3-regular plane graphs with given face sizes**

  For this generator one can specify which face sizes are allowed in the graph. So in fact fullerenes are a special case of this class. The reason to offer Fullerenes separately is to provide special options that are not implemented in this more general generator. The default is to give a set of allowed face sizes and the number of vertices. Then all plane graphs will be generated that have only faces with a size in the set of allowed faces.

  Lower and upper bounds for the numbers of faces of a given size can be supplied as an option. E.g. a lower bound of 1 forces a face of this size to be present in every graph generated.

  Furthermore the connectivity of the graphs can be chosen. The default is to generate

only 3-connected graphs.

In this part two different generators are used. The algorithm implemented in the program *CGF* is actually capable of handling/generating the more general class of graphs on arbitrary oriented surfaces of given genus. It is described in Thomas Harmuth's PhD thesis [7]. A preliminary version with the same basic ideas but differences in detail (and restricted to the plane) is described in [8]. This algorithm is optimized for relatively few allowed face sizes and is only applied for these cases and cases where additional options are chosen. In cases where a lot of face sizes are allowed (and no additional options used) the generator *plantri_ad* described in [9] is used.

- **Bipartite 3-regular plane graphs**

The user has to choose the number of vertices and a lower bound for the cyclic connectivity.

The main generator used in this part is called *plantri*. *Plantri* is in fact a program containing various different generators with different construction and isomorphismn rejection routines. The algorithms implemented in *plantri* are described in [10]. For cyclically 1- or 2-connected graphs *CGF* is used by allowing all possible even face sizes. Note that cubic graphs are cyclically 1- resp. 2-connected if and only if they are 1- resp. 2-connected. Using *CGF* for these purposes is not very efficient, because *CGF* is designed for relatively few face sizes and writing a plantri-like generator also for this part will be future work.

- **General 3-regular plane graphs**

This generator is mainly designed to generate cubic 3-connected plane graphs without strong restrictions on the face sizes as fast as possible. The same classes could be generated with the generator that accepts lists of allowed faces – one would just have to allow every face except maybe 3- and 4-gons – but this generator is much faster for the cases that can be chosen in this window and offers different options to restrict the connectivity.

One has to choose the number of vertices, the minimum face size (at least 3 – no double edge allowed) and a lower bound for the cyclic connectivity. Note that for $1 \leq c \leq 3$ a cubic graph is c-connected if and only if it is cyclically c-connected.

As an option one can choose that the given cyclic connectivity is not interpreted as a lower bound but as the exact value of the cyclic-connectivity number.

In this part *plantri* is also used.

4-regular plane graphs

- **4-regular plane graphs with given face sizes**

  Just like for the case of 3-regular plane graphs with given face sizes one can specify which face sizes may occur in the generated graphs.

  The algorithm used in this part is called *quad_restrict* and is described in [11]. It is based on *plantri* and uses an additional branch and bound criterion, similar to [9]. It generates only 3-connected graphs and is designed for the case of relatively many face sizes allowed. It does not allow extra options.

  For the future we hope to be able to reimplement the generator for 4-regular graphs described in [8]. This would allow a more efficient generation in the case of few face sizes and also more options for the structure of the graphs.

- **General 4-regular plane graphs**

  In this section one of the generators in *plantri* is used to generate all 4-regular plane graphs with a given number of vertices.

  Furthermore one of the following three options must be chosen:

  – 3-connected

  – 4-edge-connected

  – 3-connected and cyclically 6-edge connected

Planar Polycyclic Hydrocarbons

In this section planar polycyclic hydrocarbons can be generated – or to be precise: graphs in which all vertices are of degree 2 or 3 and all faces except for one distinguished outer face are pentagons or hexagons. All vertices of degree 2 must be in the outer face. In the chemical context the vertices of degree 2 are carbon atoms bonded with an additional hydrogen atom making the total degree 3.

- **Planar polycyclic hydrocarbons by formula**

  For this generator one must choose the chemical formula (e.g. $C_{31}H_{17}$) and the number p of pentagons as input (with $0 \leq p \leq 5$). Results from [12] about maximum and minimum values for the number of carbon atoms for a given number of hydrogens and pentagons are already used at this level to detect cases where no structures exist.

  Options allow the restriction to IPR structures and structures that are strictly peri-condensed. Furthermore an upper bound on the number of 3-valent vertices between two successive 2-valent vertices in the boundary can be given. This can restrict the local curvature of the boundary and interpreting the structures as realized in the plane with – more or less – equal bond lengths and equal angles around a vertex it can avoid overlaps of faces not too far apart in the boundary.

  The structures can be output with or without hydrogens (vertices with degree 1) attached to the carbon atoms (vertices) of degree 2.

  The algorithm implemented in this generator has not been published so far, but of course it has been tested against several other programs for similar tasks and data published in the literature.

- **Planar polycyclic hydrocarbons by boundary formula**

  The boundary structure of a planar polycyclic hydrocarbon can be described by the cyclic sequence of its boundary degrees. The boundary structure of azulene would e.g. be described by the cyclic sequence “222322223”. This generator generates

all planar polycyclic hydrocarbons for a given cyclic sequence of twos and threes that correspond to $0 \leq p \leq 5$ pentagons. If d stands for the number of vertices of degree two in the boundary and t for the number of vertices of degree 3 then we have $p = 6 - (d - t)$, so the number of pentagons is uniquely determined by the boundary. The restriction $p < 6$ is essential because for $p \geq 6$ the number of structures can be infinite (see e.g. nanotubes).

An option for the generator is to allow only structures with isolated pentagons and again one can choose to output structures with or without hydrogens attached.

The algorithm implemented in this generator is described in [13].

- **Planar polycyclic hydrocarbons by number of hexagons**

This generator is specialized on the efficient generation of planar polycyclic hydrocarbons with a given number of hexagons and no pentagons. The possible numbers of vertices (C-atoms) that can occur for a given number of hexagons in a benzenoid was already determined in [14]. In [12] this was generalized to the larger class of fusenes – that is the case where the structure can not necessarily be embedded in the hexagonal lattice without overlap.

As options one can restrict the generation to benzenoids, catacondensed fusenes, and Kekulean fusenes (fusenes with a perfect matching).

The algorithm implemented in this generator is described in [15] with an addition for the generation of only Kekulean fusenes described in [16].

Nanotubes and Nanocones

A nanotube is a fullerene that has two caps containing 6 pentagons each and a tube body consisting of hexagons only. Normally one does not speak of a nanotube unless the number of vertices in the tube body is extremely large compared to the number of vertices in the caps. If one wants to study the influence of the caps, one just looks at one *"half"* of the structure – that is one of the caps containing 6 pentagons together with a very long tube of hexagons. In fact this tube is considered to be infinitely long. In *CaGe* *"nanotubes"* are such one-sided/single-capped, infinite structures, or equivalently: plane

cubic graphs with 6 pentagons and all the rest hexagons. If realized with (almost) equal bond lengths the 6 pentagons of a nanotube cap force the remainder that contains only hexagons to be bent so much that it has a constant diameter – it is in fact a tube. In case of an infinite 3-regular graph with $1 \leq p \leq 5$ pentagons (and the rest hexagons) the curvature is not enough to form a tube – the part containing only hexagons becomes a cone.

The generators in this part produce finite graphs that describe the nanotubes and nanocones in a unique way.

- **Nanotubes**

  Nanotubes are classified according to some unified structure of a cycle with the 6 pentagons in its interior [17]. This interior can be described as a planar polycyclic hydrocarbon with all bounded faces pentagons and hexagons and boundary structure $(23)^k(32)^l$. This structure automatically implies that there are exactly 6 pentagons. Let us call such structures *patches*.

  The parameters k, l are used to classify the type of the nanotube. In case a patch has only hexagons neighbouring the boundary, these hexagons can be removed with the result a smaller patch with the same boundary structure. If there is at least one pentagon in the boundary, the patch is called a *cap*.

  This generator takes the parameters k, l as input and generates caps representing isomorphism classes of nanotubes with these parameters. For each isomorphism class exactly one cap representating it is generated.

  Note that a cap represents a unique nanotube (extending it to a nanotube can only be done in one way), but several caps that are not isomorphic as plane graphs can represent the same nanotube. In order to generate only representations of pairwise non-isomorphic nanotubes, only a subset of all caps for given parameters k, l are generated – one for each nanotube.

  Options are:

  – generate only caps representing IPR nanotubes

- add a given number of hexagon rings to the generated caps

- **Nanocones**

  Similar to nanotubes also nanocones can be classified by a certain boundary structure (see [18]). If $1 \leq p \leq 5$ is the number of pentagons and we set $s = 6-p$ then for $p \in \{1, 5\}$ one can always find a closed cycle so that the inner part has a boundary structure of $((23)^k 2)^s$ for some k. For $p \in \{2, 3, 4\}$ the boundary structure is either $((23)^k 2)^s$ or $((23)^k 2)^{s-1}(23)^{k-1}2$. The patches with all maximal 23-subsequences of the boundary of the same length are called symmetric and the others are called nonsymmetric. Patches with one of these boundary structures and a pentagon in the boundary are called nanocone-caps. If a patch has the given boundary structure but no pentagon in the boundary, then all faces adjacent to the boundary can be removed with the result a patch of the same *type* (symmetric or unsymmetric) and the same parameter s but the new parameter k' with $k' = k - 1$.

  In this part one first chooses whether one wants a symmetric or a nonsymmetric boundary. Then one chooses the number p of pentagons in the cone and the *length of the longest side* – which is the parameter k in the description of the boundary above.

  The generator generates exactly one representative for every isomorphism class of nanocones with the given parameters.

  Options are:

  - generate only caps representing IPR nanocones
  - add a given number of hexagon rings to the generated caps

The algorithm implemented in this generator is described in [19].

Triangulations

This section is for graphs that may be more interesting from a mathematical than a chemical point of view (except for their duals). The two main classes are *triangulations of the plane* – that is plane graphs where every face is a triangle – and *triangulations of the*

disk – that is plane graphs with one marked face (that is to be considered the unbounded outer face) where all bounded faces are triangles.

- **Triangulations of the plane**

  A triangulation of the plane is a plane graph where all faces are 3-gons. Triangulations are duals of cubic plane graphs and in fact some of the generators for cubic graphs are generators for triangulations where afterwards the dual is taken. Similarly, here some generators are generators for cubic graphs with afterwards the dual taken. The reason to have the graph and the dual in separate parts is just for easier usage. Note also that in case of graphs that are not 3-connected, the dual may be a multigraph and multigraphs are not supported by *CaGe*. So some of the options that can be used for the graph are not present for the dual.

  This section contains three subsections:

  – **General triangulations**

    The main parameter is the number of vertices.

    The options are

    * One can restrict the minimal degree. The default is 3 which allows all simple triangulations.
    * One can give a lower bound for the connectivity. The default is 3 which allows all simple triangulations.
    * One can choose that only graphs are output where the connectivity number is equal to the lower bound.

  – **Triangulations with given vertex degrees**

    This is the dual case to *3-regular plane graphs with given face sizes*.

    One must give the number of vertices and a list of allowed vertex degrees.

    As an option one can give upper and lower limits for the number of vertices with each given degree.

– **Eulerian triangulations**

Eulerian triangulations are triangulations where all vertex degrees are even – or equivalently: triangulations that have a Eulerian cycle

The main parameter is the number of vertices. Due to the Euler formula the smallest degree in a Eulerian triangulation is always 4.

The options are

* One can give a lower bound for the connectivity. The default is 3 which allows all simple triangulations.
* One can choose that only graphs are output where the connectivity number is equal to the lower bound. While in case the chosen lower bound is 4 this is automatically the case, in case of 3 it forces non-facial triangles in the output graph.

- **Triangulations of the disk**

In cases where the disk – the outer face – has 3 edges in the boundary, the graph is isomorphic to a triangulation of the plane. But as the disk is considered to have a special status, a fixed triangulation of the plane occurs as often as a triangulation of the disk as there are orbits of the automorphism group on the triangles. In case of 2-dimensional drawings, the outer face is displayed as the outer face in the drawing (see details of the output routines later). The option to compute and display 3-dimensional embeddings is not disabled for these structures, but one must take into account that the outer face plays no special role in these embeddings, so triangulations of the disk that are isomorphic as graphs or even as plane graphs can no longer be distinguished.

The main parameter that has to be chosen is the number of vertices. The options are

– One can fix the number of vertices on the boundary of the disk.

– One can forbid, allow or even require chords – that is edges between vertices on the boundary that are not part of the boundary.

- One can allow or forbid 2-valent vertices on the boundary. Note that in the interior no 2-valent vertices can occur because they would require double edges in order to form a triangulation.

The algorithms implemented for these generators are described in [10].

Quadrangulations

Quadrangulations are plane graphs where the boundary of every face is a 4-cycle. Quadrangulations are duals of 4-regular graphs and the generators used here are the same as in the part for 4-regular graphs. There are two main sections:

- **General quadrangulations:**

  The main invariant one has to give is the number of vertices. Furthermore one of the following 4 classes has to be chosen:

  - quadrangulations with minimum degree 2 (for at least 4 vertices these are all simple quadrangulations)
  - quadrangulations with minimum degree 3
  - 3-connected quadrangulations
  - 3-connected quadrangulations without 4-cycles that are not the boundary of a face

- **Quadrangulations with given vertex degrees:**

  In this part one can choose the number of vertices and the set of allowed vertex degrees. The generator is the same as in the part *"4-regular plane graphs with given face sizes"* – only that the dual graphs are displayed (which are also 3-connected). This means that also for this part a reimplementation of [8] would be important to speed up the generation in cases where only few vertex degrees are allowed.

General plane graphs

In this section one can generate and display plane graphs without too many restrictions. The main invariant is again the number of vertices.

The options are

- a lower bound between 1 and 5 for the minimum degree.
- a lower bound between 1 and 3 for the connectivity of the graph.
- lower and upper bounds for the number of edges.
- an upper bound on the face size.

In case of 3-connected graphs one can choose to output the duals of the graphs described by these parameters. In this case the parameters change their meaning – the number of vertices becomes the number of faces, the minimum degree becomes the minimum face size and the maximum face size becomes an upper bound on the vertex degree. The meaning of the limits for the number of edges doesn't change.

The algorithm implemented for this generator is based on the triangulation generator and is described in [10].

External Generator

In the section *external generator* it is possible to read graphs from a file or a generator for plane graphs that is not contained in *CaGe*. The output has to be in a format understood by *CaGe* though. For details on the format see [5].

Output options

After having chosen the class of graphs to be generated and clicking on *next*, the window shown in Fig 5 appears and one has to choose what to do with the generated graphs.

Though several options in the generators allow to restrict the classes of graphs that are generated, the user may have requirements that are even more special and only a superclass of the class of interest can be found. One may e.g. be interested in fullerenes that do not only obey the IPR rule, but where each pair of pentagons is separated by

at least 2 hexagons. For this purpose the pre-filter option is included. If the option is switched on, one can type in the name of a program that is used to filter the output. The graphs generated are piped into this filter program and the graphs written by the filter are used as input for further processing as e.g. display. The program must read and write a format described on [5]. *CaGe* then behaves as if only those graphs written by the filter program were generated.

Output options allow to draw the graphs in 2 and 3 dimensions. The algorithms used to draw the graphs are described in [20].

The following options are offered for the output:

3D: The graphs are embedded and drawn in 3 dimensions. The embedding routine is designed for interactive use of *CaGe*, so speed was the highest priority. The embeddings are designed to have – as far as possible – equal bond lengths and equal angles around the vertices. For fullerene-like structures this normally gives an embedding on a topological ball, but for other structures – e.g. triangulations with vertices of high degree – these restrictions may lead to highly entangled 3-dimensional representations. In the future, specialized embedders for these classes will have to be developed. Though designed mostly for speed, in case of chemically reasonable structures, like e.g. fullerenes, the 3-dimensional embeddings turned out to be good enough as starting points for molecular dynamics programs. The 3D embeddings can be visualized with Rasmol (see [21]), Jmol (see [22], Figure 8) and can be written to a file or a pipe. The information written to a file or pipe contains the adjacency information of the graph and the 3-dimensional coordinates of the vertices.

One can also compute an unfolding of the structures in order to build 3D models (see Figures 9,10). In order to get an unfolding of a structure, the button in the navigation window shown in Figure 6 must be clicked while the structure is displayed.

2D: The graphs can also be embedded and drawn in 2 dimensions. In case of fullerenes and other polyhedra, the drawing is in the form of a Schlegel diagram. The result can be viewed with a 2D viewer included in *CaGe* (see figure 7), in which case

the user can choose the outer face by clicking on the face and export the drawing as Postscript. Furthermore the embedding can be written to a file or piped into another program. The information written contains the adjacency information of the graph and the 2-dimensional coordinates of the vertices.

Adjacency information: The adjacency information of the generated graphs can also be sent to a file or a pipe without embedding the graph in two or 3 dimensions first.

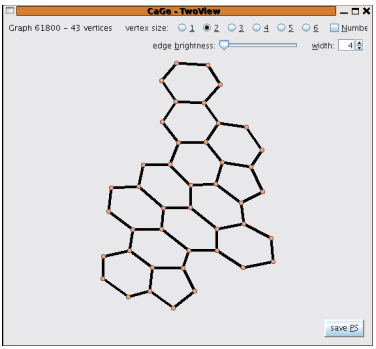

Figure 7: A 2-dimensional drawing of a planar polycyclic hydrocarbon.

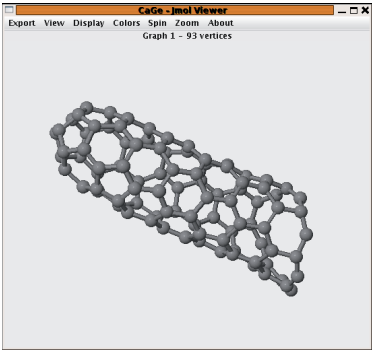

Figure 8: A 3-dimensional embedding of a partial nanotube.

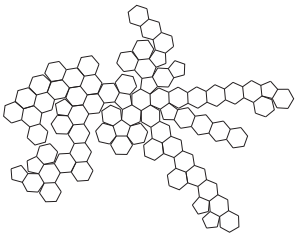

Figure 9: An unfolding of a fullerene with 140 vertices

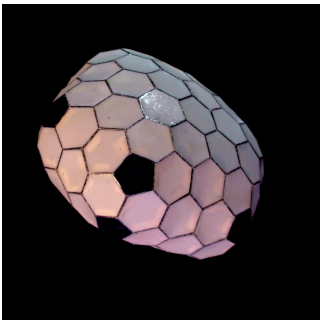

Figure 10: A model of the smallest IPR fullerene (206 atoms) without a spiral starting at a pentagon. The model is built from an unfolding.

Future work

In the future more classes of graphs will be added, but the most important work at the moment is to improve the way some graphs are drawn. The embedders in *CaGe* were originally developed for Fullerene-like structures and forcing embeddings with approximately equal bond lengths and bond angles. In some cases that differ a lot from Fullerenes (with e.g. low connectivity or vertices with high degree) this does not lead to good 2D or 3D drawings. Improving the drawing routines will have the highest priority for the near future.

Acknowledgements

A lot of people contributed to *CaGe* that are not mentioned in the list of authors of this article. We especially want to thank Thomas Harmuth, Alexander Lust, Brendan McKay, and Ulrike von Nathusius for their contributions, help and advice.

References

[1] G. Brinkmann, O. Delgado Friedrichs, A. Dress, T. Harmuth, CaGe – a virtual environment for studying some special classes of large molecules, *MATCH Commun. Math. Comput. Chem.* **36** (1997) 233–237.

[2] S. Schein, T. Friedrich. A geometric constraint, the head-to-tail exclusion rule, may be the basis for the isolated-pentagon rule in fullerenes with more than 60 vertices, *Proc. Natl. Acad. Sci. USA* **105** (2008) 19142–19147.

[3] H. Zettergren, G. Sanchez, S. Diaz-Tendero, M. Alcami, F. Martin, Theoretical study of the stability of multiply charged c-70 fullerenes, *J. Chem. Phys.* **127** (2007) 104308.

[4] P. W. Fowler, P. Hansen, D. Stevanović, A note on the smallest eigenvalue of fullerenes, *MATCH Commun. Math. Comput. Chem.* **48** (2003) 37–48.

[5] Homepages of *CaGe*: `http://www.mathematik.uni-bielefeld.de/~CaGe/` `http://caagt.ugent.be/CaGe/`.

[6] G. Brinkmann, A. W. M. Dress, A constructive enumeration of fullerenes, *J. Algorithms* **23** (1997) 345–358.

[7] T. Harmuth, *The Construction of Cubic Maps on Orientable Surfaces*. PhD thesis, Universität Bielefeld (2000).

[8] G. Brinkmann, O. Heidemeier, T. Harmuth, The construction of cubic and quartic planar maps with prescribed face degrees, *Discr. Appl. Math.* **128** (2003) 541–554.

[9] G. Brinkmann, B. D. McKay, U. von Nathusius, Backtrack search and look-ahead for the construction of planar cubic graphs with restricted face sizes, *MATCH Commun. Math. Comput. Chem.* **48** (2003) 163–177.

[10] G. Brinkmann, B. D. McKay, Fast generation of planar graphs, *MATCH Commun. Math. Comput. Chem.* **58** (2007) 323–357; see http://cs.anu.edu.au-/~bdm/index.html.

[11] S. Funke, Erzeugung dreizusammenhängender Quadrangulierungen mit eingeschränkten Knotengraden, Master's thesis, Universität Bielefeld (2004).

[12] J. Bornhöft, G. Brinkmann, J. Greinus, Pentagon-hexagon-patches with short boundaries, *Eur. J. Combin.* **24** (2003) 517–529.

[13] G. Brinkmann, B. Coppens, An efficient algorithm for the generation of planar polycyclic hydrocarbons with a given boundary, *MATCH Commun. Math. Comput. Chem.* **62** (2009) 209–220.

[14] F. Harary, H. Harborth, Extremal animals, *J. Comb. Inf. Syst. Sci.* **1** (1976) 1–8.

[15] G. Brinkmann, G. Caporossi, P. Hansen, A constructive enumeration of fusenes and benzenoids, *J. Algorithms* **45** (2002) 155–166.

[16] G. Brinkmann, C. Grothaus, I. Gutman, Fusenes and benzenoids with perfect matchings, *J. Math. Chem.* **42** (2007) 909–924.

[17] M. S. Dresselhaus, G. Dresselhaus, P. C. Eklund, *Science of Fullerenes and Carbon Nanotubes*, Academic Press, 1996.

[18] D.J. Klein, Topo-combinatoric categorization of quasi-local graphitic defects, *Phys. Chem. Chem. Phys.* **4** (2002) 2099–2110.

[19] G. Brinkmann, N. Van Cleemput, Classification and generation of nanocones, manuscript.

[20] O. Delgado Friedrichs, Fast embeddings for planar molecular graphs, in: P. Hansen, P. Fowler, M. Zheng (Eds.), *Discrete Mathematical Chemistry*, Am. Math. Soc., Providence (2000), pp. 85–95.

[21] Homepage of rasmol and openrasmol:. http://rasmol.org/.

[22] Jmol: an open-source java viewer for chemical structures in 3d; http://jmol.sourceforge.net/.