Bulletin UASVM, Horticulture 65(2)/2008
pISSN 1843-5254; eISSN 1843-5394

CVNET SOFTWARE PROGRAM AND GENERATED STRUCTURES

Monica FODOR, Monica STEFU, Liliana FODOR, T. RUSU

Technical University Cluj-Napoca, 103-105 B-dul Muncii, Cluj-Napoca
mstefu02@yahoo.com

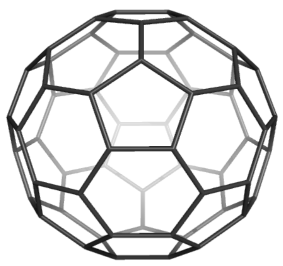
Keywords: algorithms for map operations, surface covering, nanostructures

Abstract: A map M is a combinatorial representation of a graph embedded on a surface and creates a tessellation on that surface. The map operations are topological transformations of a given map. A computer program, CVNET, was made for generating maps or coverings representing fullerenes/ hyper structures, by map operations. In this paper are presented algorithms for map operations, which analyze the original graph and then generate the new ones. Some examples of generated objects by map operations are illustrated at the end.

INTRODUCTION

The Fullerenes are molecules composed entirely of carbon of valence 3, in the form of a hollow sphere, ellipsoid, tube or their derivations and combinations[i], with different coverings. A classical fullerene called also buckyball has a pseudo-spherical shape made up entirely of 12 pentagons and various numbers of hexagons (Figure 1 a), as results from the Euler's formulas (see below). Cylindrical fullerenes called carbon nanotubes are constructed mostly from hexagons[ii], but they often contain defects as non hexagonal polygons (Figure 1 b).

C_{60} classical fullerene (a)



Armchair Polyhex Nanotubes (b)

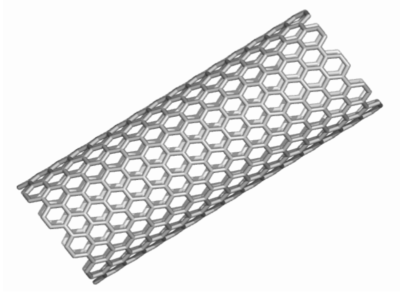


Figure 1

A buckyball fullerene (a) and a nanotube (b)

A graph is said to be *embedded* in a surface *S* when it is drawn on *S* so that no two edges intersect[iii]. A graph is planar if it can be embedded in a plane. A *map M* is a combinatorial representation of a graph embedded on a surface[iv] and creates a tessellation on that surface. The *map operations* are geometrical and topological transformations of a given map. Like the graph notations, we denote in a map: *v* – the number of vertices, *e* – the number of edges, *f* – the number of faces and *d* – the vertex degree. There are some basic relations in a map[v]:

$$\sum dv_d = \sum sf_s = 2e$$

where v_d is the number of vertices of degree *d* and f_s the number of faces with s edges, respectively. The two relations are joined in the Euler's formula[vi]:

$$v - e + f = \chi(M) = 2(1-g)$$

where χ is the *Euler's characteristic* and *g* the genus[3] of the graph (*i. e.*, $g = 0$ for a planar or spherical graph and 1 for a toroidal graph). Positive/negative χ values indicate positive/

negative curvature of a map or lattice. This formula is useful for checking the consistency of an assumed structure. For example, the C_{60} fullerene has vertices of degree 3 v_3=60, edges e=90 and faces of size 5 and 6 f_5=12, f_6=20. For molecular modeling, the fullerenes are represented by molecular graphs embedded in surfaces or corresponding maps. Thus the surfaces are covered with diverse patterns by the polygons or faces of the maps, obtaining hyper structures. As theoretical models, fullerenes with faces of various sizes and coverings have been proposed[vii,viii]. Some of them are obtained by map operations or by joining fullerenes fragments[ix].

A computer program, CVNET[x], with options for map operations was made. A few map operations implemented in the CVNET program are presented in the following section. Section Algorithms used for Map Operations describes and presents some of the algorithms used by the CVNET program.

MATERIAL AND METHODS

The *Map Operations* can be classified in simple, composite, generalized and others operations[2,7]. *Dualization Du* of a map is built as follows: locate a point in the centre of each face and join two such points if their corresponding faces share a common edge. The new map is called the *dual Du*(*M*). The following relations exist between the elements of the initial and resulting map[4]:

$$Du(M): \qquad v = f_0\,;\ \ e = e_0\,;\ \ f = v_0$$

where subscript index "0" mark the corresponding parameters in the parent map.

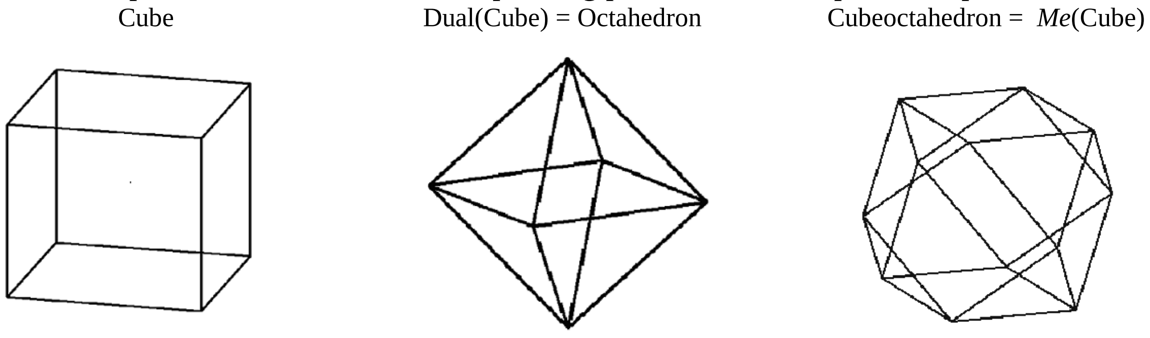


Figure 2
Results of the Dual and Medial operations on the Cube

Medial Me[xi] puts the new vertices as the midpoints of the original edges and join two vertices if and only if the original edges span an angle on a face in the original map. The resulting parameters are:

$$Me(M): \qquad v = e_0\,;\ \ e = 2e_0\,;\ \ f = f_0 + v_0$$

Figure 2 illustrates Dual and Medial operations on the Cube. Others simple operations are Truncation, Stellation, P_4 Capping, and P_5.

The *composite operations*[xii] can be obtained by sequences of some simple operations. The vertex multiplication factor $m = v/v_0$ for a composed or generalized operation that can be denoted by (a,b)[xiii], applied on a 3-valent map can be counted by the Goldberg's relation[xiv]:

$$m = (a^2 + ab + b^2);\ a \ge b;\ a + b > 0$$

The *Quadrupling Q*[11,xv] and the *Capra Ca*[xvi] operations on the Cube are illustrated in the Figure 3. The multiplication factors for trivalent maps are $m(2,0)$ = 4 and $m(2,1)$ =7, respectively. Capra is a chiral operation (has two variants) and generates chiral objects.

Q(Cube) (a) Ca(Cube) (b)

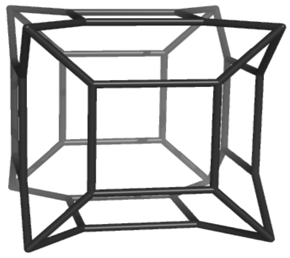
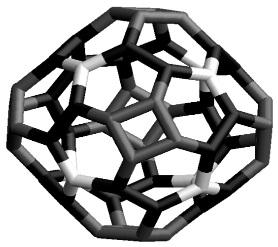



Figure 3

Quadrupling (a) and Capra (b) operations on the Cube

Other chiral operations are some *generalized operations*[13] noted by (a,b), where $a \neq b$ and $a \neq 0$. The (3,1) generalized operation (Figure 4) has the multiplication factor m=9+3+1=13.

$(3,1)_1$ (Cube) $(3,1)_2$ (Cube)

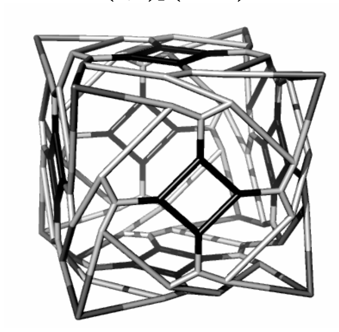
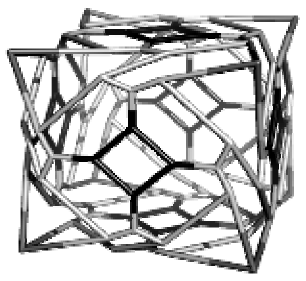



Figure 4

(3,1) operation has two variants, resulting in chiral objects.

The (a,b) operations, where $a = b$ or $a = 0$, generates achiral objects. Some of the generalized operations can be obtained by successions of composite operations.

Supra-Coverings[7]

A perfect Clar structure[xvii,xviii] PC is a disjoint set of faces, built up on all vertices in M, whose boundaries form a regular 2-valent spanning subgraph. A PC structure is associated with a Fries structure[xix], which is a spanning subgraph of the initial structure from which the edges from PC structure have been eliminated. A trivalent polyhedral graph, like that of fullerenes, has a PC structure if and only if it has a Fries structure[11]. Leapfrog Le is the only operation that provides PC results. Figure 5 presents Perfect Clar and Fries structures of C_{180}.

$Le(C_{60})$ = Le(Le(C_{20})) = (3,0) (C_{20}) = C_{180}, PC structure (a) $Le(C_{60})$ =C_{180}, Fries structure (b) $Q(C_{60}) = Q(Le(C_{20})) = C_{240}$, Pcor flowers (c)

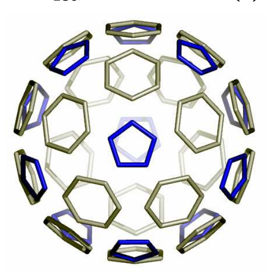
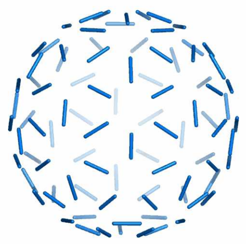
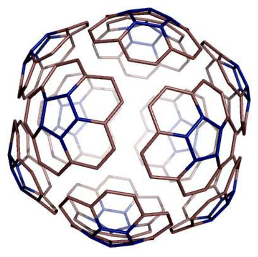




Figure 5

Perfect Clar PC (a) and Fries structures of C_{180} (b); Perfect Coranulenic PCor structure of C_{240} (c)

By extension, a coranulenic system[8] was considered (Figure 5 c). A Perfect Coranulenic structure PCor is a disjoint set of (supra) faces covering all vertices in the molecular graph. The operation sequence $Le(Q(M)) = Q(Le(M))$, that is equivalent to the (2,2) generalized operation, provide a PCor structure. The PCor transform superimposes over PC, thus, any

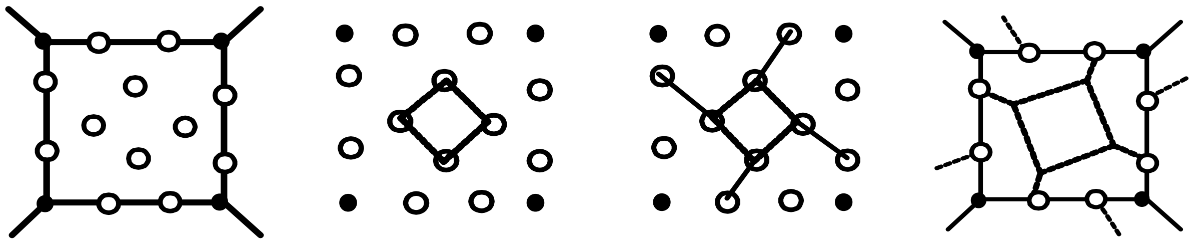
PCor is necessarily a PC. The PC and PCor structure is expected to contribute to the stability of the whole molecule. Others operations, like Capra, Disjoint Azulenic operation and Rotated Disjoint Azulenic operation generates others kinds of "flowers".

Algorithms used for Map Operations

CageVersatile .Net CVNET software program, written in C#, under .NET Framework, is a program that has options for map operations and it generates closed or open lattices covering nanostructures. The input and output files are in HyperChem – .hin format and contain the representation of labelled graphs[xx].

First, the program finds the rings or the faces of the input graph and then it realizes the map operations on every face. To find the faces in a map, the program uses a recursive algorithm to detect all the cycles of the graph with length less or equal to a input parameter. In this respect, the program uses the table of links between vertices from the input file. Then the program selects the cycles as *faces*, in ascending order of their size, under the condition that *any edge shares maximum 2 faces*. Next, the algorithm of map operations works in the following steps, for every face or cycle:

1. Put the vertices of the resulting map at their coordinates, calculated function of the coordinates of the vertices from the initial map, for every old face.
2. Link the vertices located inside of an old face.
3. Link the vertices located in different old faces. In case of chiral resulting objects, the old faces are visited by adjacency, so that the proper links are accomplished (Figure 6).



Step 1. Put the new vertices — Step 2. Link the inner vertices — Step 3. Link the above linked vertices with the external or border vertices

The algorithm steps of the Capra operation on a square face Figure 6

We detail below the general algorithm that creates the new map on the basis of the old faces, used at composite, generalised and others complex operations. The variable *c* is a three-dimensional array, containing the faces of the graph. In the array *c*, *c*[*i*,0,0] is the number of faces contained in the vector *c*[*i*], which begins at the vertex *i*; *nv* = *c*[*i*,*j*,0] is the number of vertices of the face *c*[*i*,*j*], while *c*[*i*,*j*,1] ... *c*[*i*,*j*,*nv*] is the list of vertices of the face *c*[*i*,*j*]. The variable *n* stores the number of vertices in the input map. Similary like *c*, *m* stores the list of edges of the initial molecular graph and some additional new vertices, used within operations; *m*[*i*,*j*] represents an edge with the end-points in *m*[*i*,*j*,1] and *m*[*i*,*j*2]. The variable *cip* is an array that will store the resulting graph, with vertices, their coordinates and the links between vertices.

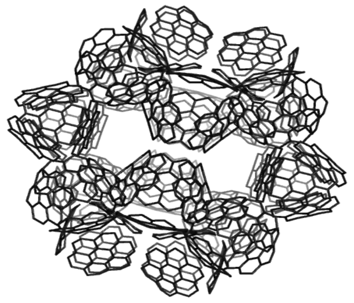
```
for i := 1 to n
  for j := 1 to c[i,0,0]
    nv := c[i,j,0];
    coord_center(c[i,j], cc);            // the array cc will store the 3D coordinates (x,y,z) of the center of the face
    for k := 1 to nv
          v_next := (k mod nv) +1;  // the vertex following vertex no. k in the face
    //add the new vertices that are inside the face and between the center and the vertices c[i,j,k], c[i,j,v_next]
          add_vertices_inside(c[i,j,k], c[i,j,v_next], cc, cip);
    endfor k
    for k := 1 to nv
         v_next := (k mod nv) +1;   // link the vertices that are between the vertices c[i,j,k], c[i,j,v_next]
         link_vertices_inside(c[i,j,k], c[i,j,v_next], cip);
    endfor k
   endfor j
  endfor i
  for i := 1 to n
   for j := 1 to m[i,0]
    add_vertices(m,i,j,cip);  // adds the rest of the border vertices that are close to the edge m[i,j]
   endfor j                   // and link them if possible
  endfor i                    //in case of chiral operation, visit the faces by adjacency and link the border vertices,
                              // otherwise the algorithm stops here.
  nc := 0; c[i,j,nv+2] :=0;  // nc numbers the visited faces;  all the faces are marked with "0", as non -visited.
  do while nc < nci          // do while exists a face not visited; nci is the number of faces in the original graph.
   for i := 1 to n           // take all the faces in the array c
    for j := 1 to c[i,0]
     nv := c[i,j,0];
     if (nc = 0) or ((c[i,j,nv+2] =0) and (AlreadyVisited(c,i,j,m)) then
      // if it is the first face or the current unvisited face shares an edge with an already visited face
      nc := nc + 1;          // visit the face i.e. link the border vertices from different faces
      c[i,j,nv+2] := 10;     // mark the face as visited
                             // the border vertices from the adjacent visited face can link in 2 ways
      link := verify_linking_mode(c,i,j,m);
      for k := 1 to nv
         add_border _links(link,c,i,j,k,m,cip); // add links to the border vertices from the current face
      endfor k
     endif
    endfor j
   endfor i
  loop
  WriteF(cip);               // writes the array cip into the output file.
```

For map operations that generates supra-coverings "flowers", the program has options to generate the whole map or/and only the "flowers" structure and eventually also the complementary structure. After generating the whole structure, it is optimized to get a rounded shape, and then coordinates of the whole structure can be transferred to the structure of "flowers" to get the same shape. For any object, it can be checked if it has a Perfect Clar PC or Perfect Coranulenic PCor structure.

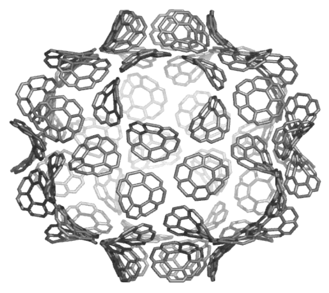
RESULTS AND DISCUSIONS

A large variety of objects of different genuses and coverings can be obtained by successions of map operations. In the following figures are presented some examples.

(2,2) operation on a structure starting from tetrahedrons



(2,2) (Op(Ca(C_{20}))) g=6



Disjoint Corannulenic Structures (PCor) obtained by faces opening (Op), starting from Dodecahedron/C_{20} Figure 7

(2,2) (Le(Op(Ca(Tetrahedron)))) S_2 (S_2(Op(Ca(Cube)))) S_2 (S_2 (S_2(C_{20})))

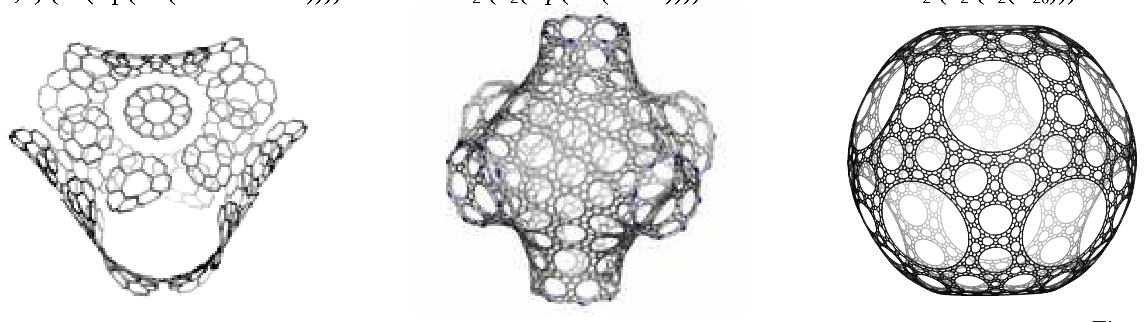


Figure 8

Other objects obtained by map operations

The program works on any faces sizes and vertices degree. There is an option to consider all the cycles that have the length less or equal the input parameter, not only the faces. Full objects can be generated in this way.

CONCLUSIONS

Various molecular graphs/maps/hyper structures, with diverse tiling and genuses, were generated by CVNET computer program for map operations. They were optimised and were further topological analysed from stability point of view.

REFERENCES

1. http://en.wikipedia.org/wiki/Fullerene, http://en.wikipedia.org/wiki/Fullerene#Variations
2. Diudea, M. V., P. E. John, 2001,Covering polyhedral tori, Commun. Math. Comput. Chem. (MATCH), 44, p103-116.
3. Harary, F., 1969, Graph Theory, Addison - Wesley, Reading, M. A..
4. Pisanski, T., M. Randić, 2000, in Geometry at Work, M. A. A. Notes, 53, p.174-194.
5. Euler, L., 1736, Solutio Problematis ad Geometriam Situs Pertinentis. Comment. Acad. Sci. I. Petropolitanae 8, p.128 – 140
6. Euler, L., 1758, Elementa doctrinae solidorum et demonstratio nonnularum insignium proprietatum quibus solida heddris planis inclusa sunt praedita, Novi Comment. Acad. Sci. I. Petropolitanae, 4, p. 109-160.
7. Diudea, M. V., 2005, Covering Nanostructures, In: M. V. Diudea, Ed., Nanostructures-Novel Architecture, NOVA, New York, p. 203-242.
8. Diudea, M. V., 2004, Covering Forms in Nanostructures, Forma (Tokyo), 19 (3), p. 131-163.
9. Diudea, M. V., C. L. Nagy, 2007, Periodic Nanostructures, Ed. Springer, Netherlands
10. Stefu, Monica, M. V. Diudea, 2007, CageVersatile (.Net), "Babes-Bolyai" University, Cluj.
11. Fowler, P. W. and T. Pisanski, 1994, J. Chem. Soc. Faraday Trans. 90, 2865-2871.
12. Diudea, M. V., P. E. John, A. Graovac, M. Primorac, T. Pisanski, 2003, Leapfrog and Related Operations on Toroidal Fullerenes. Croat. Chem. Acta, 76, p. 153-159.
13. Diudea, M. V., Monica Stefu, P. E. John, A. Graovac, 2006, Generalized operations on maps, Croat. Chem. Acta, 79, p. 355-362.
14. Goldberg, M., 1937, Tohoku Math. J. 43, p. 104-108.
15. Fowler, P. W., J. E. Cremona, J. I. Steer, 1988, Theor. Chim. Acta, 73, 1.
16. Diudea, M. V., 2003, Capra – a Leapfrog related map operation, Studia Universitatis Babeş-Bolyai, Chemia, XLVIII, 2, p. 3-16.
17. Clar, E., 1964, Polycyclic Hydrocarbons, Acad. Press, London.
18. Clar, E., 1972, The Aromatic Sextet, Wiley, New York.
19. Fries, K., J. 1927, Liebigs Ann. Chem., 454, p. 121-324.
20. Stefu, Monica, Daniela Butyka, M. V. Diudea, L. Jantschi, B. Parv, 2005, Algorithms for basic operations on maps, In: M. V. Diudea, Ed., Nanostructures-Novel Architecture, NOVA, New York, p. 243-267.