Exhaustive Combinatorial Enumeration

Mathieu Dutour Sikirić

Rudjer Bošković Institute and

Nagoya University

January 16, 2010

I. The problem and the algorithm

Combinatorial enumeration

- Example of problem considered:
  - List all 3-valent plane graphs with faces of gonality 5 and 9 and all 9-gonal faces in pairs.
  - List all cliques of 600-cell.
  - List all triangulations of the sphere on n vertices.
  - List all isohedral (r, q)-polycycles.
- The main feature of the proposed problems is that we do not have any intelligent way of doing it.
- We do not want only enumeration, we want to have those objects so as to work with them.

Limitation of hope

- In the best scenario, the speed of computers multiply by 2 every year.
- In most combinatorial problems, the number of solutions grow much more than exponentially in the size of the problem.
- One typical example is the listing of all graphs with n vertices:
  - The number of labeled graphs with n vertices is $2^{\frac{n(n-1)}{2}}$.
  - The symmetric group $Sym(n)$ act on those labeled graphs.
  - So, the number of unlabeled graph is around

$$2^{\frac{n(n-1)}{2}} \frac{1}{n!} \simeq \sqrt{2}^{n^2}$$

- So, the progress brought by computer should diminish as time goes on.

II. The automorphism and isomorphism problems

The graph isomorphism problem

- Suppose that we have a graph G on n vertices $\{1, \ldots, n\}$, we want to compute its automorphism group $Aut(G)$.
  g is formed of all elements in $Sym(n)$ such that

$$\{g(i), g(j)\} \in E(G) \text{ if and only if } \{i, j\} \in E(G)$$

- Suppose that G_1 and G_2 are two graphs on n vertices $\{1, \ldots, n\}$, we want to test if G_1 and G_2 are isomorphic, i.e. if there is $g \in Sym(n)$ such that

$$\{g(i), g(j)\} \in E(G_1) \text{ if and only if } \{i, j\} \in E(G_2)$$

- It is generally believed that those problems admit solution in a time bounded by a polynomial in n.

The program nauty

- The program nauty of Brendan McKay solves the graph isomorphism and the automorphism problems.

  `http://cs.anu.edu.au/people/bdm/nauty/`
- nauty is extremely efficient in doing those computations.
- nauty can deal with directed graph but this is not recommended.
- nauty can deal with vertex colors.
- nauty iterates over all $n!$ permutation but it prunes the search tree so as to obtain a fast running time.
- nauty has no problem at all for graph with several hundred vertices.

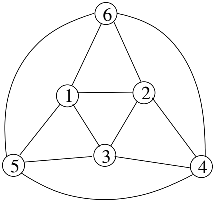
The reduction to a graph

Why focus on graph?

- We have many other combinatorial problems:
  - subset of vertex-set of a graph,
  - set system,
  - edge weighted graph,
  - plane graph,
  - partially ordered set, etc.
- If M is a "combinatorial structure", then we define a graph $G(M)$, such that:
  - If M_1 and M_2 are two "combinatorial structure", then M_1 and M_2 are isomorphic if and only if $G(M_1)$ and $G(M_2)$ are isomorphic.
  - If M is a "combinatorial structure", then $Aut(M)$ is isomorphic to $Aut(G(M))$.

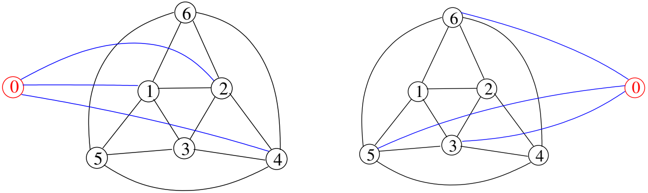
Subset of vertex-set of a graph

- Suppose that we have a graph G, two subsets S_1, S_2 of G, we want to know if there is an automorphism ϕ of G such that $\phi(S_1) = S_2$.



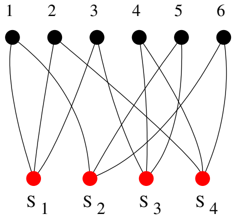
$$S_1 = \{1, 2, 4\}$$
$$S_2 = \{3, 5, 6\}$$

- The method is to define two graphs associated to it:



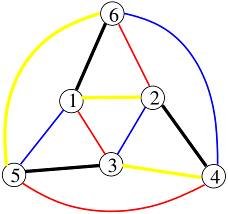
Set systems

- Suppose we have some subsets $S_1, \ldots, S_r$ of $\{1, \ldots, n\}$. We want to find the permutations of $\{1, \ldots, n\}$, which permutes the S_i.
- We define a graph with $n + r$ vertices j and S_i with j adjacent to S_i if and only if $j \in S_i$
- Example $\mathcal{S} = \{\{1, 2, 3\}, \{1, 5, 6\}, \{3, 4, 5\}, \{2, 4, 6\}\}$:



Edge colored graphs

- G is a graph with vertex-set $(v_i)_{1\leq i\leq N}$, edges are colored with k colors $C_1, \dots, C_k$:

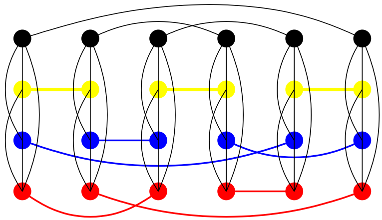


- We want to find automorphisms preserving the graph and the edge colors.
- We form the graph with vertex-set (v_i, C_j) and
  - edges between (v_i, C_j) and $(v_i, C_{j'})$
  - edges between (v_i, C_j) and $(v_{i'}, C_j)$ if there is an edge between v_i and $v_{i'}$ of color C_j

  We get a graph with kN vertices.

Edge colored graph

- The picture obtained is:

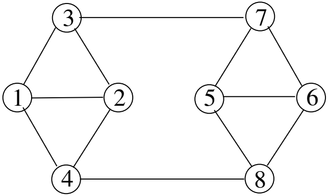


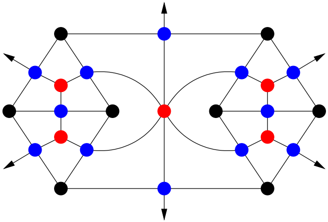
- Actually, one can do better, if the binary expression of j is $b_1 \dots b_r$ with $b_i = 0$ or 1 then we form the graph with vertex-set (v_i, l), $1 \leq l \leq r$ and
  - edges between (v_i, l) and (v_i, l')
  - edges between (v_i, l) and $(v_{i'}, l)$ if the binary number b_l of the expression of C_j is 1.

  This makes a graph with $\lceil \log_2(k) \rceil N$ vertices.

Plane graphs

- If G is a simple 3-connected plane graph then the skeleton determine the embedding, we can forget the faces.
- If G has multiple edge and/or is not 3-connected we consider the graph formed by its vertices, edges and faces with adjacency given by incidence





- This idea extends to partially ordered sets, face lattices, etc.

Canonical form

- `nauty` has yet another wonderful feature: it can compute a canonical form of a given graph.
- One possible canonical form of a graph is obtained by taking the lexicographic minimum of all possible adjacency matrix of a given graph.
- This canonical form is not the one used by `nauty` though I don't know which one is used.
- Suppose that one has N different graphs from which we want to select the non-isomorphic ones.
  - if one do isomorphism tests with `nauty` then at worst we have $\frac{N(N-1)}{2}$ tests.
  - If one computes canonical forms, then we have N calls to `nauty` and then string equality tests.
- This is a key to many computer enumeration goals.

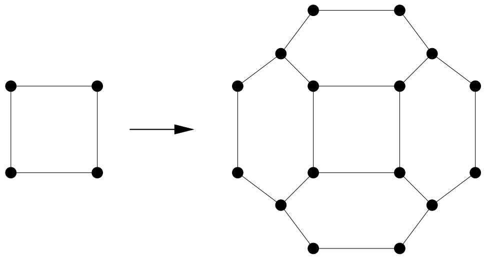
Conclusion

- Computing the automorphism group of a given combinatorial structure is not difficult.
- The only difficulty is that one has to be careful in defining the graph $G(M)$.
  - For example if K_n is the complete graph with edge colors, then the line graph $L(K_n)$ is a vertex colored graph
  - But $|Aut(K_4)| = 4!$ and $|Aut(L(K_4))| = 2 \times 4!$
- In many cases, most of the time is taken by the slow program writing the graph to a file.

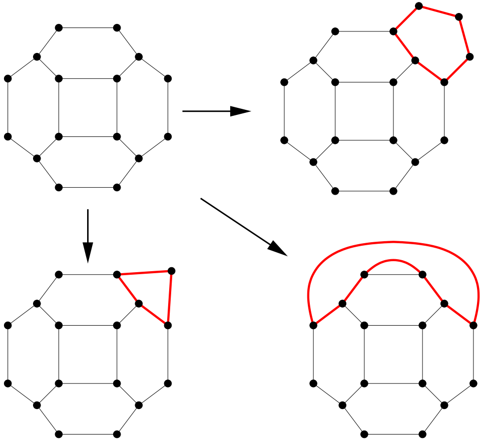
III. Exhaustive enumeration

Plane graph example

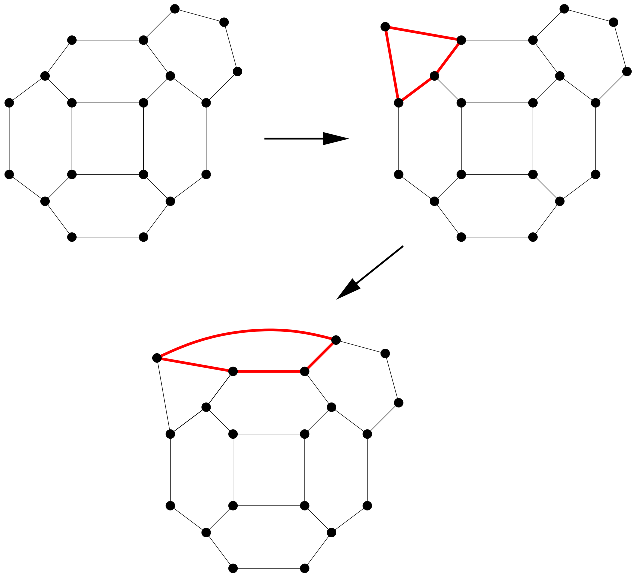
- A $(\{a, b\}, k)$-graph is a k-valent plane graph, whose faces have size a or b.
- A $(\{a, b\}, k)$-graph is called:
  - aR_i if every a-gonal face is adjacent to exactly i a-gonal faces
  - bR_j if every b-gonal face is adjacent to exactly j b-gonal faces
- Suppose that one wants to enumerate the $(\{4, 6\}, 3)$-graphs, which are $4R_0$ and $6R_3$
- We start with a single 4-gon



Next step

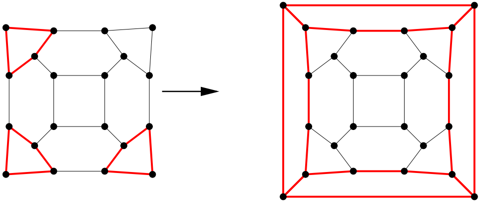


Next step

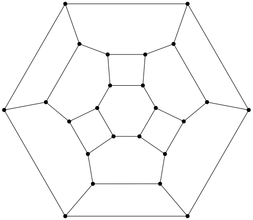


Conclusion of the process

- So, we are left with



- In the end we obtain the following graph:



24, O_h

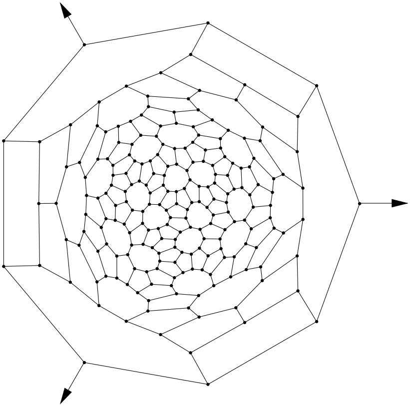
Features of the exhaustive method

- We have a lot of intermediate steps, even if in the end we obtain a few or no objects.
  - For example the enumeration of $(\{4, 9\}, 3)$-graphs, which are $4R_1$, $9R_4$ took several days with in the end no graph found.
- The time run is unpredictable.
- The symmetry and the feature of the obtained objects cannot be used in their determination.
- At every step we have several possibilities. We need to make some choices.
- The method is essentially a computerized case by case analysis. But the program is actually more stupid than us and a priori it cannot do generalizations easily.

- Sometimes, we run into infinite loops with a non-terminating program even if the finiteness is proved beforehand.
  - All $(\{3,4\},4)$-graphs, which are $3R_0$ and $4R_3$ have 30 vertices
  - The program find those graphs but actually it continues with some partial structure of more than 30 vertices.
- A key point is some pruning functions with which one can prove that a structure admits no extension
  - All $(\{4,8\},3)$-graphs, which are $8R_4$ satisfy to
    - $e_{4-4} = 12$ with e_{4-4} the number of edges separating two 4-gons
    - $x_0 + x_3 = 8$ with x_i the number of vertices contained in i 4-gons
  - So, if $e_{4-4} > 12$ or $x_0 + x_3 > 8$, then we can discard this case.
- If we have several possible options, select the one with the minimal number of possibilities of extension.
- After the first stages, the speedup obtained by isomorphism rejection decrease and can result in a slow down.

A successful example

- We wanted to determine the $(\{4,9\},3)$-graphs $9R_1$
- With an exhaustive enumeration scheme, it took less than 1 hour, 21 graphs were generated, the largest of which is



212, T

IV. Augmentation schemes (or orderly generation)

Computing cliques up to symmetry

- If G is a graph, then a k-clique is a set S of k vertices such that any two elements in S are adjacent.
- We want to enumerate all cliques up to symmetry of the graph G, not just maximal cliques.
- The straightforward algorithm is the following:
  - Take the list of all k-cliques up to isomorphism.
  - For every k-clique, consider all possibilities of adding a vertex to it, i.e. the $(k+1)$-cliques it is included in.
  - Reduce by isomorphism this set of $(k+1)$-cliques.

  Iterate from 1 to the clique number of the graph.
- The problem is that one has to store the list of all k-cliques in memory.

Canonical augmentation for cliques

- We number the vertices of G. If $S \subset \{1, \dots, n\}$ then its canonical form is the lexicographic minimum of its orbit under $Aut(G)$.
- Suppose that $S = \{x_1, \dots, x_{k-1}, x_k\}$ is a lexicographically minimal k-clique. Then the subset

$$S' = \{x_1, \dots, x_{k-1}\}$$

  is a $(k-1)$-clique, which is lexicographically minimal.
- The method is then the following
  - Take the list of k-cliques, which are lexicographically minimal.
  - For every lexicographically minimal k-clique $S = \{x_1, \dots, x_k\}$, consider all its extensions

$$S'' = \{x_1, \dots, x_k, t\} \text{ with } x_k < t$$

    and select the $(k+1)$-cliques amongst them, which are lexicographically minimal.

Feature of this scheme

- For every lexicographically minimal k-clique $S_k = \{x_1, \ldots, x_k\}$, we have a canonical path to obtain it:

$$\begin{array}{rcl} S_1 & = & \{x_1\} \\ S_2 & = & \{x_1, x_2\} \\ \vdots & & \\ S_{k-1} & = & \{x_1, x_2, \ldots, x_{k-1}\} \end{array}$$

- The memory is no longer a problem.
- This method split extremely well on parallel or cluster computers.
- It is difficult to make this kind of schemes, you have to create a canonical way to construct a structure.

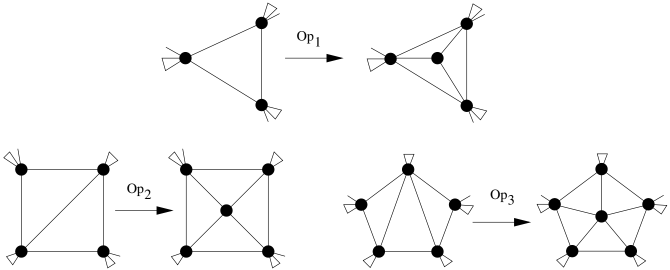
All cliques of 600-cells

- 600-cell has 120 vertices, and a symmetry group H_4 of size 14400.
- The cliques of the complement of 600-cell correspond to some polytopes, whose faces are regular polytopes.

k	nb						
1	1	7	334380	13	74619659	19	25265
2	7	8	1826415	14	54482049	20	1683
3	39	9	7355498	15	26749384	21	86
4	436	10	21671527	16	8690111	22	9
5	4776	11	46176020	17	1856685	23	1
6	45775	12	70145269	18	263268	24	1

Generating triangulations

- A triangulation of the sphere is a plane graph whose faces are 3-gons.
- Triangulations are generated by iteration of the following operations starting from the Tetrahedron:

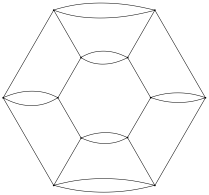
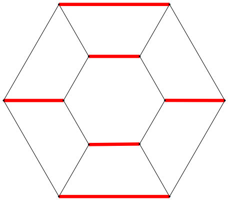


- One can get a canonical path leading to a given triangulations.
- This method is used by the program `plantri` of Gunnar Brinkmann and Brendan McKay.

V. The homomorphism principle

An example

- Suppose that one wants to generate 4-valent plane graphs with faces of size 2, 4, 6 such that every vertex is contained in exactly one face of size 2

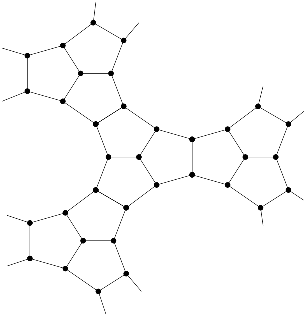



- If one collapse the 2-gons to edges, one obtains a $(\{4,6\},3)$-sphere. The 2-gons correspond to a perfect matching in it.
- The method is then
  - List all $(\{4,6\},3)$-graphs
  - For every $(\{4,6\},3)$-graph, list its perfect matching.

  We factorize the difficulties.

Isohedral (r, q)-polycycles

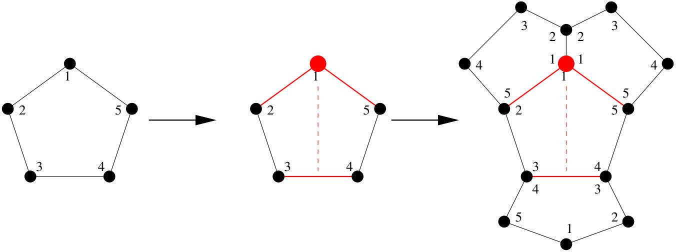
- A (r, q)-polycycle is a plane graph, whose interior faces are r-gons and all vertices are of degree q except those on the boundary, which have degree in $[2, q]$.
- It is isohedral if its symmetry group act transitively on the r-gonal faces. Below is an isohedral $(5, 3)$-polycycle



- By the isohedrality, we simply need to define the image along the edges of an r-gon

The method used

- If we have the r-gon, then we first specify:
  - the edges which are boundary edges,
  - the vertices which are interior vertices,
  - the stabilizer of the r-gon.
- Then we enumerate all possibilities around all interior edges.
- One example:



Enumeration results

$r \downarrow q \rightarrow$	3	4	5	6	7	8
3	2	3	4	4	4	4
4	3	6	9	11	11	13
5	7	17	24	38	37	51
6	12	45	67	130	123	196
7	28	157	257	518	452	896
8	58	486	894	2095	1781	3823

Number of isohedral (r, q)-polycycle for $r, q \leq 8$.

Some references

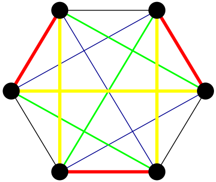
- P. Kaski, P. Ostergard, *Classification algorithms for codes and designs*, Springer Verlag 2006.
- P. Ostergard, *Constructing combinatorial objects via cliques*, Surveys in combinatorics 2005, 57–82, London Math. Soc. Lecture Note Ser., 327.
- G. Brinkmann and B.D. McKay. *Fast generation of planar graphs*, to appear in *MATCH Commun. Math. Comput. Chem.*
- B. McKay, *Isomorph-free exhaustive generation*, J. Algorithms, **26** (1998) 306–324.
- G. Brinkmann, *Isomorphism rejection in structure generation programs*, Discrete mathematical chemistry, DIMACS Ser. Discrete Math. Theoret. Comput. Sci. **51** (Amer. Math. Soc., 2000) 25–38.

Some references

- A. Kerber, *Applied finite group actions*, 2^{nd} edition, Springer-Verlag, 1999.
- B. McKay, *Practical graph isomorphism*, Congressus Numerantium **30** (1981) 45–87.
- C.J. Colbourn, R.C. Read, *Orderly algorithms for graph generation*, Internat. J. Comput. Math. **7-3** (1979) 167–172.
- R.C. Read, *Every one a winner or how to avoid isomorphism search when cataloguing combinatorial configurations*, Algorithmic aspects of combinatorics (Conf., Vancouver Island, B.C., 1976). Ann. Discrete Math. **2** (1978) 107–120.
- I.A. Faradzev, *Generation of nonisomorphic graphs with a given distribution of the degrees of vertices* (Russian) Algorithmic studies in combinatorics "Nauka", Moscow, **185** (1978) 11–19.

1-factorizations of K_{2n}

- A 1-factor of K_{2n} is a set of $2n-1$ perfect matchings in K_{2n}, which partition the edge-set of K_{2n}.
- The graph K_6 has exactly one 1-factorization with symmetry group $Sym(5)$, i.e. the group $Sym(5)$ acts on 6 elements.



graph	$\lvert$isomorphism types$\rvert$	authors
K_6	1	
K_8	6	1906, Dickson, Safford
K_{10}	396	1973, Gelling
K_{12}	526915620	1993, Dinitz, Garnick, McKay

THANK

YOU