MATCH
Communications in Mathematical and in Computer Chemistry

MATCH Commun. Math. Comput. Chem. **86** (2021) 453-488

ISSN 0340 - 6253

On the Counting Complexity of Mathematical Nanosciences

Juan Andrés Montoya

Universidad Nacional de Colombia, Departamento de Matemáticas

jamontoyaa@unal.edu.co

(Received January 26, 2021)

Abstract

We investigate the algorithmic hardness of a series of counting problems that come from crystal physics and fullerene chemistry. We claim that those problems are representative of mathematical nanosciences, and we observe that all them are sparse. It follows from Mahaney's work that sparse problems cannot be hard for NP. Then, we have to use a different complexity class to analyze the aforementioned (seemingly hard) problems. We study the complexity class $\#\mathrm{P}_1$, which is constituted by all the tally counting problems that belong to the counting class #P.

We conjecture that counting matchings in square grids, counting Hamiltonian cycles in square grids, and counting Clar sets in fullerene graphs are all hard for $\#\mathrm{P}_1$. We prove some weak results related to these conjectures. We also consider the restriction of these three problems to carbon nanotubes, and we prove that those restrictions can be solved in logarithmic time. We get these tractability results about nanotubes as corollaries of a Büchi-like Theorem for linear lattices.

We study counting (and optimization) problems that come from crystal physics and fullerene chemistry. We use the term nanosciences to refer those two research areas as well as some other closely related areas. A key feature of those counting problems is that all them are sparse. This implies that those problems cannot be hard for NP. However, it seems that some of them are hard. We ask: for which complexity class are those problems hard?

1 Introduction

Let us introduce the counting problems that are studied in this paper, as well as some of the tools that are used to study those problems.

1.1 Sparse counting problems

There are many counting problems coming from different areas of the natural sciences. Let us just mention the counting of spanning subtrees, perfect matchings, Eulerian orientations, matchings, self-avoiding walks and Hamiltonian circuits (see [1], [2] and the references therein). The first three problems motivated the invention of counting algorithms that are landmarks in the area, namely: Kirchhoff's Matrix-Tree Algorithm [3], Kasteleyn's Algorithm [4] and the Transfer Matrix Counting Method [5]. The last three problems, in turn, have resisted all the attacks: we do not know how to efficiently count those objects.

Valiant studied the counting of perfect matchings in bipartite graphs, and he proved that this problem is hard for a class that he called #P [6]. The complexity class #P allows the analysis of counting problems in NP. It is known that counting perfect matchings, counting matchings, counting self-avoiding walks (SAW's, for short), counting Eulerian orientations and counting Hamiltonian circuits are all #P-hard problems [1]. It is also known that any #P-hard problem is hard for the polynomial hierarchy [7]. On the other hand, Kasteleyn proved that one can count perfect matchings of planar graphs in polynomial time [4]. Thus, we get that the restriction to planar graphs makes some hard problems become tractable. However, we have to observe that there are many problems that remain #P-hard when restricted to planar graphs. This is the case with the counting of matchings, Hamiltonian paths, SAW's and Eulerian orientations [1]. We claim that those hardness results for planar graphs are not the end of the investigations on the algorithmic hardness of the latter four problems: nanoscientists use to be interested in highly homogeneous graphs that we call *nanostructures*. The canonical example of nanostructures are *square grids*. It could be argued that the following four problems, besides the counting of perfect matchings, are the most studied counting problems coming from the mathematical nanosciences:

1. Counting Hamiltonian circuits in square grids.
2. Counting matchings in square grids.

3. Counting SAW's in square grids.

4. Counting Eulerian orientations in square grids with periodic boundary conditions.

We have to observe that the restriction to square grids can have a positive effect on the algorithmic hardness of the above problems: the counting of Eulerian orientations becomes tractable when restricted to square grids with periodic boundary conditions [5]. However, it seems that the remaining three problems remain hard when we restrict them to square grids. Let us use the symbols $\#_1 Ham$, $\#_1 Matchings$ and $\#_1 SAW$ to denote those problems. We ask: what can be said about the algorithmic hardness of $\#_1 Ham$, $\#_1 Matchings$ and $\#_1 SAW$? To begin with, it should be said that those problems have resisted all the attacks, and, as far as we know, nobody knows either a polynomial time solution or a hardness proof for one of them. We would like to analyze the algorithmic hardness of $\#_1 Ham$, $\#_1 Matchings$ and $\#_1 SAW$, as well as the algorithmic hardness of a fourth problem coming from fullerene chemistry (see below). One of our goals is to find an explanation for more than seventy years of failures trying to count Hamiltonian circuits, matchings and SAW's.

Remark 1 *The latter assertion holds for exact counting. The problems $\#_1 Ham$, $\#_1 Matchings$ and $\#_1 SAW$ admit full polynomial time randomized approximation schemes [8].*

Remark 2 *The problems $\#_1 Ham$ and $\#_1 SAW$ are closely related: Hamiltonian circuits are a special type of self-avoiding walks. Then, it makes sense if we focus on the problems $\#_1 Ham$ and $\#_1 Matchings$.*

1.2 Sparse problems and the class $\#\mathbf{P}_1$

Is $\#_1 Ham$ hard for #P? Is $\#_1 Matchings$ hard for #P? We have to observe that $\#_1 Ham$ and $\#_1 Matchings$ are *sparse problems*.

Definition 3 *We say that a problem L is sparse, if and only if, problem L has polynomial many instances of each possible size.*

Suppose that P is different of NP, and let L be a sparse problem in NP. We have that L cannot be NP-hard. This is the famous theorem of Mahaney [9]. It should be noted that

Mahaney's argument does not actually require the sparse language to be in NP: there is a sparse problem that is NP-hard, if and only if, P = NP. Further, E $\neq$ NE, if and only if, there exist sparse languages in NP that are not in P [10].

Definition 4 *A tally counting problem is a function $f : \mathbb{N} \to \mathbb{N}$. We say that f is a tally counting problem in NP, if and only if, there exists a nondeterministic polynomial time unary Turing machine $\mathcal{M}$ such that for all $n \geq 1$ the equality*

$$f(n) = \#acc_{\mathcal{M}}(1^n)$$

holds, where the symbol $\#acc_{\mathcal{M}}(1^n)$ denotes the number of accepting computations of $\mathcal{M}$, on the input 1^n.

Assume that deterministic exponential time is different to nondeterministic exponential time (E $\neq$ NE). We get that there must exist tally counting problems in NP that are not in P. Then, we can dream of proving that $\#_1 Ham$ and $\#_1 Matchings$ are not in P. How can we achieve this? By proving that those two problems are hard for a suitable complexity class. We think that this suitable complexity class is the class $\#\mathrm{P}_1$ introduced by Valiant and defined below [11].

Definition 5 *A counting problem f belongs to $\#P_1$, if and only if, f is polynomial time reducible to a tally counting problem in NP.*

What do we know about the class $\#P_1$?

Theorem 6 *We have:*

1. *$\#P_1$ is closed under polynomial time counting reductions.*
2. *$\#P_1$ contains problems that are complete under polynomial time counting reductions [11].*
3. *$\#P_1$ does not contain #P-hard problems: if $P \neq NP$, then the class $\#P_1$ cannot contain problems that are hard for NP [10].*
4. *Assume that E is different to NE, then the class $\#P_1$ contains problems that are not in P.*

We think that we can use the class $\#\mathrm{P}_1$ to successfully analyze the problems $\#_1 Ham$ and $\#_1 Matchings$: if we could prove that one of those problems is $\#\mathrm{P}_1$-hard, then we could use items 3 and 4, of the above list, to explain more than seventy years of failures trying to count Hamiltonian circuits and matchings in square grids.

1.3 Complete problems for $\#\mathbf{P_1}$

Let us begin with a quotation:

> The Class $\#\mathrm{P}_1$ is not nearly as interesting as #P due to the absence of any interesting natural problem known to be complete in it.

This quotation is taken from Rudich's lecture notes on Computational Complexity Theory [12]. We could think that Rudich's remark is unfair, given that $\#\mathrm{P}_1$ contains problems that are complete for this class. However, we have to understand that Rudich's remark is pointing out an important question:

All the hardness results about the class $\#P_1$ are generic results, which assert that certain large sets of counting problems contain some unspecified problems that are hard for $\#P_1$. None of those results tells apart a single specific problem to claim that this problem is hard for this complexity class.

The consequence of all this is that we do not count with a problem like SAT, which is complete for NP, or a problem like #SAT, which is complete for #P. Or, more concretely: we do not know how to use the class $\#\mathrm{P}_1$ when we want to analyze specific problems like $\#_1 Ham$ and $\#_1 Matchings$. We think, together with Rudich, that the construction of a natural $\#\mathrm{P}_1$-complete problem is an important open problem related to the complexity theory of tally counting problems.

1.4 Tractable problems: One–dimensional structures

Our main goal is to study the algorithmic problems that come from mathematical nanosciences. We would like to establish positive results concerning the tractability of some large families of those problems. We conjecture that most counting problems related to square grids are intractable. We believe that this conjectured intractability of square grids relies on the two-dimensional character of those graphs: the set of square grids is a set of unbounded *treewidth*. We also believe that most counting problems about fullerene are intractable: the set of fullerene graphs is also a set of unbounded treewidth.

We believe, on the other hand, that one-dimensional homogeneous structures such as carbon nanotubes are highly tractable. We investigate the tractability of counting and optimization problems that can be defined by monadic second order formulas. We prove that all those problems can be solved in logarithmic time over the Word–Level RAM model when those problems are restricted to tally sets of one-dimensional homogeneous structures. Then, we get that we can count matchings, SAW's, Hamiltonian circuits and Clar sets of carbon nanotubes in logarithmic time.

1.5 Organization of the work and contributions

This work is organized into five sections including the introduction.

In section 2 we study two problems that come from crystal physics, namely the problems $\#_1 Ham$ and $\#_1 Matchings$. We prove some weak results related to the (likely) $\#\mathrm{P}_1$-hardness of both problems: we prove that the counting of matchings and the counting of Hamiltonian cycles become $\#\mathrm{P}_1$-hard when restricted to sparse sets that are constituted by *fuzzy* versions of the set of square grids.

In section 3 we study some counting problems that are related to fullerene molecules. We begin observing that the set of fullerene graphs is a sparse set. We conclude that our counting problems about fullerene belong to the class $\#\mathrm{P}_1$. We study in depth the problem of counting Clar sets of fullerene graphs, and we prove that a fuzzy version of this problem is hard for $\#\mathrm{P}_1$.

In section 4 we focus on Carbon nanotubes. We prove that all the counting problems studied in previous sections can be solved in logarithmic time when restricted to nanotubes and similar structures. We get those results as corollaries of the following algorithmic meta-theorem: any problem about linear lattices that can be defined by a monadic second order formula can be solved in time $O(\log(n))$.

We finish in section 5 with some few concluding remarks.

2 Crystal nano-physics: Square grids

The counting problems $\#_1 Ham$ or $\#_1 Matchings$ appear in statistical physics as the partition functions of different models of crystal growth [4]. The quasiperiodic structure of those planar lattices make them suitable models of crystal structures.

We would like to prove that $\#_1 Ham$ or $\#_1 Matchings$ are hard for $\#\mathrm{P}_1$. This is a very

hard piece of work, and it becomes harder when we realize that:

1. Most combinatorial gadgets produce graphs that are far apart from being square grids.
2. The set of square grids is a rigid set that is quickly *overflowed* for any useful combinatorial reduction.
3. Most tally sets of molecular graphs are as rigid as the set of square grids.

It seems that any hardness result for $\#\mathrm{P}_1$ implies a certain amount of uncertainty: there is a sentence, there is a census function, there is Turing machine, there is a pattern... There is a sparse set of graphs for which the problem f is $\#\mathrm{P}_1$-hard. We prove this latter type of generic results in this section: we prove that there are sparse sets of graphs that look like grids, and which are hard for the counting of Hamiltonian circuits and for the counting of matchings. More specifically, we prove that there is a sparse set of grids with holes for which the problem $\#Ham$ is $\#\mathrm{P}_1$-hard, and we prove that there is a sparse set of *hairy grids* for which the problem $\#Matching$ is $\#\mathrm{P}_1$-hard. Unfortunately, we cannot say what are those two sparse sets of fuzzy grids without running a certain algorithm on an infinite set of instances: those two hard sets of fuzzy grids are not specified by the hardness proofs.

2.1 Counting Hamiltonian circuits

A grid with holes is a graph that can be obtained from a square grid by deleting some edges (and creating some holes). Papadimitriou et al proved that it is NP-hard to recognize the grids with holes that are Hamiltonian [13]. Toda et al proved that counting Hamiltonian circuits in grids with holes is #P hard [14]. This latter result is the best evidence we have about the hardness of $\#_1 Ham$. Let $\mathcal{GH}$ be the set of grids with holes. We have to observe that $\mathcal{GH}$ is not sparse, but the reader should also observe that this set is constituted by graphs that are very close to be square grids. Moreover, as we will see, it is easy to extract a sparse subset of $\mathcal{GH}$ such that the restriction of $\#Ham$ to this subset is $\#\mathrm{P}_1$-hard.

Let f be a tally counting problem in $\#\mathrm{P}_1$ that is complete for this class. Problem f belongs to #P, and hence it can be reduced in polynomial time to $\#Ham\,[\mathcal{GH}]$. The set of queries that are made when one reduces f to $\#Ham\,[\mathcal{GH}]$ is a sparse subset of $\mathcal{GH}$, and this sparse set is $\#\mathrm{P}_1$-hard for the counting of Hamiltonian circuits.

The above idea is the key argument in the proof of the easy and weak result that we prove in this section. We should think of this basic idea as a general constructive principle: any counting problem that is hard for #P contains sparse subproblems that are hard for $\#\mathrm{P}_1$.

Definition 7 *Let L be a problem that is hard for $\#P$, let $f : \mathbb{N} \to \mathbb{N}$ be a tally counting problem that is complete for $\#P_1$, and let R be a polynomial time reduction of f in L. We say that R is tally reduction, if and only if, there exists a polynomial $p(X)$ such that given $n \geq 1$, the size of the queries that are made, when the reduction is ran on input n, is bounded below by $p(n)$.*

Remark 8 *Note that the definition of tally reduction forbids to make queries that are very small when compared with the size of the input.*

Theorem 9 *There exists a sparse subset of $\mathcal{GH}$ such that the restriction of $\#Ham[\mathcal{GH}]$ to this subset is hard for $\#P_1$.*

Proof. Let f be a tally counting problem that is complete for $\#\mathrm{P}_1$. It follows from the construction of Toda, Liskiewicz and Ogiwara that there exists a tally reduction of f in $\#Ham[\mathcal{GH}]$ (see [14]). Let R be one of those tally reductions. Given $n \geq 1$, we define $R_{f,n}$ as the set

$$\left\{ \begin{array}{c} G : G \text{ is an instance of } \#Ham[\mathcal{GH}] \text{ that occurs in a query that} \\ \text{is made when one runs } R \text{ on input } n \end{array} \right\}.$$

The set

$$\mathcal{R}_f = \bigcup_{n \geq 1} R_{f,n}$$

is an sparse subproblem of $\#Ham[\mathcal{GH}]$ that is hard for $\#\mathrm{P}_1$. ■

Remark 10 *We would like to prove that the set of square grids is hard for the counting of Hamiltonian circuits. This result seems to be out of scope. We ask: how close can we get to the set of square grids? A sparse set of square grids with holes is not an uniform set of graphs as it is the set of square grids, and we have to agree that we are far away from our target.*

We have to observe that we got the above result almost for free. We will prove a similar result concerning the counting of matchings. This latter result is more demanding and more exciting than the latter:

- We have to work harder than in the proof of the previous theorem: we have to prove, first, that the set of *hairy grids* (see below) is #P-hard for the counting of matchings.
- We prove that there exists a highly uniform sparse set of hairy grids that is hard for $\#\mathrm{P}_1$.

2.2 Counting matchings

We prove, in this section, a result about matchings that is analogous to the above result about Hamiltonian circuits: we prove that there exists a sparse set of *hairy grids* that we denote with the symbol $\mathcal{HG}_1$ and such that $\#Matchings\left[\mathcal{HG}_1\right]$ is hard for $\#\mathrm{P}_1$. We get this result as an easy corollary of the following theorem: let $\mathcal{HG}$ be the set of hairy grids, the problem $\#Matchings\left[\mathcal{HG}_1\right]$ is hard for #P. It seems that this is the best thing that we can do for the moment: to prove that a certain set of graphs that look like square grids is #P hard for the counting of the substructures under study.

Definition 11 *Let G be a rectangular grid. The graph $G^{(3)}$ is the graph that can be obtained from G after inserting two nodes on each edge of G. We say that $G^{(3)}$ is the 3-subdivision of G. A hairy grid is a quintuple $H = (G, A, B, n, m)$, such that G is a square grid; A and B are disjoint subsets of $V(G)$; and n, m are positive integers. The quadruple H corresponds to the graph that can be obtained from $G^{(3)}$ after appending paths to the nodes in $A \cup B \cup \left(V\left(G^{(3)}\right) - V(G)\right)$. The nodes in A get paths of length n, the nodes in B get paths of length m and the nodes in $\left(V\left(G^{(3)}\right) - V(G)\right)$ get nodes of length $n+1$.*

Remark 12 *Note that we do not append paths of many different lengths to the nodes of $G^{(3)}$, we append paths of only three different lengths, namely: $n, n+1$ and m.*

We use the symbol $\mathcal{HG}$ to denote the set of hairy grids. The set $\mathcal{HG}$ is not sparse. However, we can use the idea that was used in the proof of the previous result: if $\#Matchings\left[\mathcal{HG}\right]$ is #P-hard under tally reductions, then we can extract from $\mathcal{HG}$ a sparse subset that is $\#\mathrm{P}_1$-hard. This time we have to work harder, since we have to prove that $\#Matchings\left[\mathcal{HG}\right]$ is #P-hard under tally reductions.

Theorem 13 *The problem $\#Matchings\left[\mathcal{HG}\right]$ is #P-hard under tally reductions.*

Proof. Let $\#PM$ be the counting problem that consists in counting perfect matchings of bipartite graphs. Valiant proved that this problem is #P-hard [6]. We prove that $\#PM$ is polynomial time reducible to $\#Matchings\left[\mathcal{HG}\right]$.

M. Jerrum proved that there exists a polynomial time reduction of $\#PM$ in the problem $\#Matchings\left[\mathcal{P}\right]$, where $\#Matchings\left[\mathcal{P}\right]$ is the restriction of $\#Matchings$ to planar graphs (see [15]). Jerrum's reduction is the composition of two reductions. The first one is a polynomial time algorithm that computes, on input G, a planar weighted graph (G^*, ω) such that the equality

$$\#PM\left(G\right) = \sum_{\substack{M \text{ is a} \\ \text{matching of } G}} \left(\prod_{\substack{v \text{ is not} \\ \text{covered by } M}} \omega\left(v\right) \right)$$

holds, where ω is a weight function that assigns weights, within the range $\{-1, 0, 1\}$, to all the nodes in G^*. The second reduction in Jerrum's proof is used to eliminates the weights -1 and 0, and it is based on *the polynomial interpolation method* [15].

Let G be a bipartite graph, and let G^* be the output of Jerrum's first reduction. The graph G^* is a weighted planar graph that is strictly larger than G (and it implies that the reduction is tally). We have to observe that G^* could contain nodes with more than 4 neighbors. The first task to be solved is to reduce the degree of those nodes. We can use weights for this end. We can reduce the degree of any node with more than four neighbors using the following simple gadget:

Let v be a node of G^*, let $v_1, ..., v_m$ be the neighbors of v, and let $k < m - 1$. We reduce the degree of v from m to $k + 1$ by inserting two nodes v^k, w^k; deleting the edges

$$\{v, v_{k+1}\}, ..., \{v, v_m\},$$

and then adding the edges

$$\left\{v, v^k\right\}, \left\{v^k, w^k\right\}, \left\{w^k, v_{k+1}\right\}, ..., \left\{w^k, v_m\right\}.$$

The Figure 1 corresponds to the case $k = 3, m = 5$.

If we assign weights 0 and 1 to the nodes v^k, w^k, we get a planar weighted graph such that the equality

$$\sum_{\substack{M \text{ is a} \\ \text{matching of } G^{v,k}}} \left(\prod_{\substack{u \text{ is not} \\ \text{covered by } M}} \omega\left(u\right) \right) = \sum_{\substack{M \text{ is a} \\ \text{matching of } G^*}} \left(\prod_{\substack{u \text{ is not} \\ \text{covered by } M}} \omega\left(u\right) \right)$$

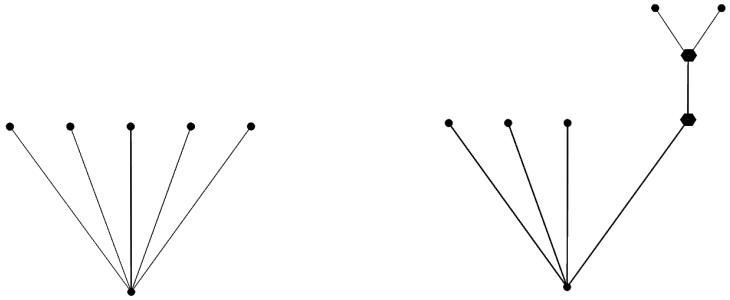

Figure 1

holds.

We can use the above gadget a polynomial number of times and obtain a planar weighted graph $G^{\$}$ such that:

1. The degree of all the nodes in $G^{\$}$ is bounded above by 4.

2. $G^{\$}$ is larger than G.

3. The equality

$$\#PM(G) = \sum_{\substack{M \text{ is a} \\ \text{matching of } G^{\$}}} \left(\prod_{\substack{u \text{ is not} \\ \text{covered by } M}} \omega(u) \right)$$

holds.

We can compute in polynomial time a subdivision of $G^{\$}$, that we denote with the symbol $G^{\#}$, and which satisfies the following:

1. Any edge of $G^{\$}$ gets subdivided into an odd number of segments (after inserting an even and nonzero number of nodes on this edge).

2. The graph $G^{\#}$ is a grid with holes, and we can compute in polynomial time a drawing of $G^{\#}$ that represents this graph as a grid with holes: we can compute in polynomial time a square grid R_G such that $G^{\#}$ is a subgraph of $R_G^{(3)}$, and we can also compute in polynomial time an embedding of $G^{\#}$ into $R_G^{(3)}$.

If we assign weight 0 to all the nodes of $G^{\#}$ that do not belong to $G^{\$}$, we get a weighted grid with holes for which the equality

$$2^{|E(G^{\$})|} \sum_{\substack{M \text{ is a} \\ \text{matching of } G^{\$}}} \left(\prod_{\substack{u \text{ is not} \\ \text{covered by } M}} \omega(v) \right) = \sum_{\substack{M \text{ is a} \\ \text{matching of } G^{\#}}} \left(\prod_{\substack{u \text{ is not} \\ \text{covered by } M}} \omega(u) \right)$$

holds.

The next step of our construction corresponds to do a little bit of sewing work: we reconstruct the cells of $R_G^{(3)}$ that are not present in the subgraph $G^{\#}$. To this end we use the following three gadgets.

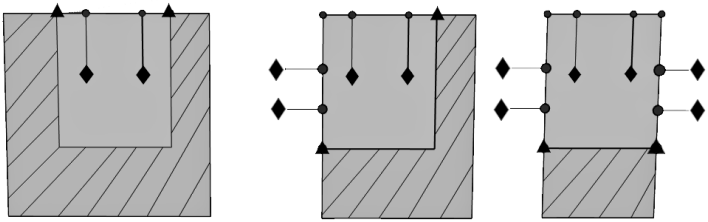

Figure 2

We can use those gadgets to reconstruct all the cells of $R_G^{(3)}$. We use the gadget on the left to reconstruct cells with one lacking edge, the gadget in the middle to reconstruct cells with two lacking edges, and the gadget on the right to reconstruct cells with three lacking edges. We assign weight zero to the nodes represented by disks and weight one to the nodes represented by diamonds. The nodes represented by triangles are the nodes that are present before the insertion of the gadget and are used to prop the construction. We leave unchanged the weights that were previously assigned to those latter nodes. At the end of the day (which lasts just polynomial time) we get the graph $R_G^{(3)}$ enriched with a weight function ω and some hairs of length 1. Let us use the symbol F_G to denote this

hairy version of $R_G^{(3)}$. We have that

$$\sum_{\substack{M \text{ is a} \\ \text{matching of } G^{\#}}} \left(\prod_{\substack{u \text{ is not} \\ \text{covered by } M}} \omega(u) \right) = \sum_{\substack{M \text{ is a} \\ \text{matching of } F_G}} \left(\prod_{\substack{u \text{ is not} \\ \text{covered by } M}} \omega(u) \right).$$

Note that $V\left(R_G^{(3)}\right)$ gets partitioned into four sets, the sets

$$\begin{aligned} A &= \left\{ v \in V\left(R_G^{(3)}\right) : \omega(v) = 0 \right\}, \\ B &= \left\{ v \in V\left(R_G^{(3)}\right) : \omega(v) = -1 \right\}, \\ C &= \left\{ v \in V\left(R_G^{(3)}\right) : v \text{ is the base node of a hair} \right\}, \\ D &= V\left(R_G^{(3)}\right) - (A \cup B \cup C). \end{aligned}$$

Let $\left(R_G^{(3)}, A, B, C\right)$ as above, and let r, s be two positive integers. We use the symbol $R_{G,A,B,C,r,s}^{(3)}$ to denote the hairy version of $R_G^{(3)}$ that is constructed as follows: we append paths of length r to all the nodes in A, paths of length s to all the nodes in B, and paths of length $r+1$ to the nodes in C. We assign weight 1 to all the nodes in this graph. We can use the polynomial interpolation method (see [15]) to compute the quantity

$$\sum_{\substack{M \text{ is a} \\ \text{matching of } F_G}} \left(\prod_{\substack{u \text{ is not} \\ \text{covered by } M}} \omega(u) \right)$$

from the data

$$\left\{ \#Matchings\left(R_{G,A,B,C,r,s}^{(3)}\right) : r, s \leq p(n) \right\},$$

where $p(X)$ is a suitable polynomial. Thus, given G, one can compute in polynomial time a square grid R_G, subsets $A, B, C \subset V\left(R_G^{(3)}\right)$ and a polynomial $p(X)$ such that $\#PM(G)$ can be easily computed from the data

$$\left\{ \#Matchings\left(R_{G,A,B,C,r,s}^{(3)}\right) : r, s \leq p(n) \right\}.$$

Altogether, we get a reduction of $\#PM$ into the problem $\#Matchings\left[\mathcal{HG}\right]$. The theorem is proved. ■

Remark 14 *The hairy grids $R_{G_n,A_n,B_n,C_n,r,s}^{(3)}$ resemble us Inca quipus. Quipus are graphs made with ropes, and which were used by Incas for accounting. It is interesting to observe that those quipu graphs were complete for the accounting needs of the Incas, while our quipus are complete for $\#P_1$.*

3 Carbon nano-chemistry: Fullerene

Square grids are examples of graphs that can be constructed by assembling together many identical (similar) pieces. There are many molecules whose molecular graphs hold this property. Let us just mention *graphene molecules*, which are planar (two-dimensional) carbon molecules that are completely constituted by carbon hexagons, that is: graphene is an allotrope of carbon consisting of a single layer of atoms arranged in a two-dimensional *honeycomb lattice*. A second allotrope of carbon that exhibits the same kind of regularity is diamond: those crystals are built by assembling several copies of the *diamond cubic structure*. We claim that mathematical nanosciences are naturally focused on this type of structures, as nanotechnology is strongly based on this type of *regular* nanomaterials.

There is a third allotrope of carbon that exhibits the same kind of regular structure, and which has been instrumental in the development of nanotechnology, we are refering to fullerene.

Definition 15 *A fullerene graph is a planar, cubic and triconnected graph such that all its faces are either hexagons or pentagons (see [16] and the references therein).*

Remark 16 *There are infinitely many fullerene graphs, and each one of them corresponds to the molecular graph of a fullerene molecule.*

3.1 Counting Clar Sets of Fullerene

It has been observed that some properties of fullerene molecules can be reasonably predicted from the values that are taken by some few topological indices (see [19] and the references therein). *The Kekule Number* and *The Clar number* are two of those indices [17].

Definition 17 *Let G be a fullerene graph, the Kekule Number of G is equal to the number of perfect matchings in G.*

We can efficiently compute the above index in the planar case. Thus, we focus on computing The Clar Number of fullerene [18]. First, some definitions.

Definition 18 *Let G be a planar graph together with a planar embedding of G that we denote with the symbol c_G. Let M be a perfect matching of G, and let F be a even-sized*

face. We say that F is a resonant face for the triple (G, c_G, M), if and only if, the equality $|F \cap M| = \frac{|F|}{2}$ holds.

Remark 19 *It is worth to remark that the notion of face is not intrinsic to G, but depends on the planar embedding c_G.*

Definition 20 *Let M be a perfect matching and let $\mathcal{F} = \{F_1, ..., F_m\}$ be a set of even-sized faces that are resonant for (G, c_G, M). We say that $\mathcal{F}$ is an independent set of resonant faces, if and only if, the faces $F_1, ..., F_m$ are pairwise node-disjoint. We say that $\{F_1, ..., F_m\}$ is a Clar set for the pair (G, c_G), if and only if, there exists a perfect matching M such that $\{F_1, ..., F_m\}$ constitutes an independent set of resonant faces for the triple (G, c_G, M).*

There exists an important connection between Clar sets and Kekulé structures. Suppose that $\{F_1, ..., F_m\}$ is an independent set of resonant faces for (G, c_G, M). Given $I \subset \{1, ..., m\}$, we can construct a perfect matching M_I as follows: for all $i \in I$ we replace the edges in $F \cap M$ by the edges in $F \cap (co\text{-}M)$. We note that given $I \neq J$, the perfect matchings M_I and M_J are different, and we also note that we can construct 2^m different perfect matchings from M and the set $\{F_1, ..., F_m\}$: if there is a large set of resonant faces for the triple (G, c_G, M), then the graph G admits a large number of Kekule structures.

Definition 21 *Let (G, c_G) be as above, and let M be a perfect matching. The Clar number of the triple (G, c_G, M) is denoted with the symbol $Clar\,(G, c_G, M)$ and is defined as*

$$\max\{|I| : \{F_i : i \in I\} \text{ is an independent set of resonant faces for this triple}\},$$

the Clar number of (G, c_G) is defined by

$$Clar\,(G, c_G) = \max\{Clar\,(G, c_G, M) : M \text{ is a perfect matching for } G\},$$

while the Clar Number of G is equal to

$$Clar\,(G) = \max\{Clar\,(G, c_G) : c_G \text{ is a planar embedding of } G\}.$$

Remark 22 *Fullerene graphs are triconnected, and it implies that all the planar embeddings of a fullerene G are equivalent. This means that the set of faces of a fullerene graph G does not depend on the planar embedding that is is chosen to represent G in the plane. We get that the Clar number of a fullerene G is the same for all the planar embeddings of this graph, and we get that this number is intrinsic to G [16].*

Consider the following optimization problem:

Problem 23 Clar

- *Input:* (G, c_G)*, where* G *is a planar graph and* c_G *is a planar embedding of* G*.*
- *Problem: compute the Clar number of* (G, c_G) *.*

The above problem is NP-hard [20]. On the other hand, the restriction of this problem to planar bipartite graphs lies in P [21]. Let us consider the counting version of $Clar$.

Problem 24 #Clar

- *Input:* (G, c_G)*, where* G *is a planar graph and* c_G *is a planar embedding of* G*.*
- *Problem: compute the number of Clar sets for the pair* (G, c_G) *, whose size is equal to* $Clar\,(G, c_G)$ *.*

We have

Theorem 25 *The problem #Clar is #P-hard.*

Proof. Let $\#IS$ be the counting problem that consists in computing the number of maximum independent sets of a graph G given as input. Let $\#IS\,[3\mathcal{P}]$ be the restriction of this problem to the set of planar graphs of maximum degree three. It is known that $\#IS\,[3\mathcal{P}]$ is #P-hard [22]. We exhibit a polynomial time counting reduction of this problem into the problem $\#Clar$. The reduction is a composition of two reductions. The first one is a reduction of $\#IS\,[3\mathcal{P}]$ into the problem $\#IS\,[\mathcal{OP}]$, where $\mathcal{OP}$ is the set constituted by all the pairs (G, c_G) such that G is a planar graph, c_G is a planar embedding of G, and all the faces in (G, c_G) have odd size. The second reduction is a reduction of the latter problem into the problem $\#Clar$. This latter reduction is essentially the same reduction used by Bernath and Berczi-Kovacz in their NP-hardness proof [20], which, as a matter of fact, is a parsimonious reduction. We focus on the first reduction.

Let G be a 3-regular planar graph. The independent sets of G are intrinsic to G and does not depend on the embedding of G. Then, we compute in polynomial time a embedding of G, and we denote this embedding with the symbol c_G. If all the faces of

(G, c_G) have odd size there is nothing to do. Then, suppose that (G, c_G) has some faces of even length. Let v a node of G, and let F be a face that contains this node. Given the pair (v, F), we append to node v a small graph $G_{v,F}$. We append this gadget in such a way that it gets completely included in the interior of F. The graph $G_{v,F}$ is one of the following two graphs.

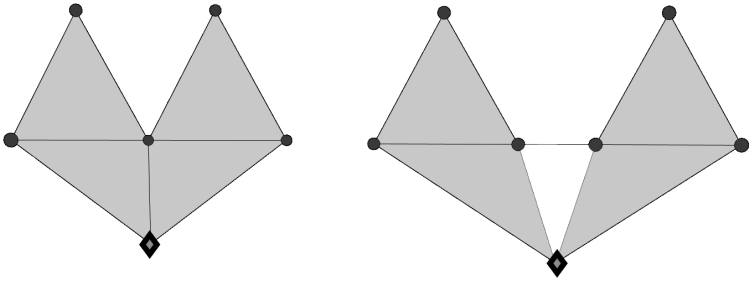

Figure 3

We insert gadget $G_{v,F}$ by identifying the node $v \in V(G)$ and the node on the bottom, which is represented by a diamond. We use the symbol A to denote the gadget on the right, and the symbol B to denote the one on the left. We insert a gadget $G_{v,F}$ for each pair (v, F). We do the latter in such a way that those small graphs do not overlap. Given a pair (v, F), we choose the gadget to be inserted according to the following rule:

If the length of F is odd, we append to each one of its nodes a copy of A. If the length of F is even, we choose exactly one node of F and we append to it a copy of B. We append a copy of A to the remaining nodes in F.

At the end of the day we get a pair (G^*, c_{G^*}), such that all the faces in (G^*, c_{G^*}) have odd size. Let I_G be the size of the maximum independent sets of G, we have that I_{G^*} is equal to $9I_G + 8(n - I_G)$. Moreover, there exists a bijective correspondence between the set maximum independent sets of G and that of G^*. We get our first reduction.

Remark. *Note that we appended a total of 3 gadgets to each node in G, and note that the graphs that belong to the range of the above reduction are 3-regular planar graphs with three different icicles (A or B) appended to each one of their nodes. Take into account that any 3-regular planar graph has a subdivision that is isomorphic to a hexagonal system with holes. We get that the graphs that belong to the range of this first reduction are hexagonal*

systems (honeycomb lattices) with holes and icicles appended to all their nodes. We say that those graphs are snowy honeycomb lattices.

Now suppose that we get an instance of $\#IS\left[\mathcal{OP}\right]$, say a pair (H, c_H). We can suppose that (H, c_H) belongs to the range of the previous reduction, and hence we can suppose that H is a planar graph that does not contain degree one nodes. We construct a suitable instance of $\#Clar$ from the pair (H, c_H). We use the symbol (H^*, c_{H^*}) to denote this instance. We proceed as follows:

Let e be an edge of H. We have that e belongs to two different faces of (H, c_H) that we denote with the symbols F_e and R_e. We add two vertices v_{F_e} and v_{R_e}, each one placed in the interior of the corresponding face. We also add the edge $\{v_{F_e}, v_{R_e}\}$. Moreover, if we are given a pair of edges, say $\{u, v\}, \{v, w\}$, and those edges belong to a common face F, we add the edge $\left\{v_{F_{\{u,v\}}}, v_{F_{v,w}}\right\}$ (see [20]).

We do the above things for any edge e, for any face F and for any pair $\{u, v\}$, $\{v, w\}$. At the end of the day we get the pair (H^*, c_{H^*}). Note that any face of (H^*, c_{H^*}) corresponds either to a face of (H, c_H), or to a vertex of H. The faces related to nodes of H are all even faces. Recall that all the faces of (H, c_H) are odd faces. We get that the even faces of (H^*, c_{H^*}) are the ones corresponding to vertices of H, and we get that the size of the maximum independent sets of H is an upper bound on The Clar Number of (H^*, c_{H^*}). The graph H^* has a perfect matching

$$M = \{\{v_{F_e}, v_{R_e}\} : e \in E\left(H\right)\}.$$

Note that any even face of (H^*, c_{H^*}) is resonant for M. We get that the Clar Number of (H^*, c_{H^*}) is equal to the size of the maximum independent sets of H. The theorem is proved. ■

We know that $\#Clar$ is $\#P$-hard under parsimonious reductions. However, we are interested in the problem $\#Clar\left[\mathcal{F}\right]$, which is the restriction of $\#Clar$ to the set of fullerene graphs. Thus, let us ask: how hard is the problem $\#Clar\left[\mathcal{F}\right]$? We have that this problem cannot be NP-hard. The latter is a consequence of Mahaney's Theorem, and the following theorem [23].

Theorem 26 *The number of fullerene graphs of size n is $O\left(n^{10}\right)$.*

We ask: is $\#Clar\left[\mathcal{F}\right]$ hard for $\#\mathrm{P}_1$?

Remark 27 *It is easy to recognize the square grids that are Hamiltonian: a square grid of size n^2 is Hamiltonian, if and only if, n is an even number. Most decision problems about square grids are equally easy and can be solved in polynomial time. We do not know of a hard problem about square grids . The problem $Clar\,[\mathcal{F}]$ seems to be hard. We think that the set $\mathcal{F}$ could be a rich source of problems that are hard for NP_1 (the decision version of $\#P_1$).*

We would like to prove that $\#Clar\,[\mathcal{F}]$ is $\#\mathrm{P}_1$-hard. The best thing that we can do, at this moment, is to prove that there exists a sparse set of *snowy honeycomb lattices* that is hard for $\#\mathrm{P}_1$. This result about snowy honeycomb lattices is analogous to some previous results discussed in this paper: there exists a tally set of decorated grids (decorated with holes, or decorated with hairs, or decorated with holes and icicles) that is hard for $\#\mathrm{P}_1$. The main question remains unsolved: can we prove the $\#\mathrm{P}_1$-hardness of a natural counting problem about grids(fullerene) without decorations?

4 Tractability

There exist organic molecules that can be represented as long chains of identical pieces that are arranged in an uniform way. We use the term linear lattices to denote those one-dimensional molecules. There are important examples of linear lattices, let us just mention the following ones: carbon nanotubes, linear polymers, helicenes, one-dimensional crystals.

4.1 Carbon nanotubes

A carbon nanotube is a cylindrical molecule obtained by rolling up a graphene sheet, which is a hexagonal grid with a carbon atom at every vertex [24].

Let M be a carbon nanotube. We can think of the molecular graph of M as a one-dimensional sequence of cylindric strips, each one constituted by the same number of hexagons. Let k be the number of hexagons in one of the strips constituting M. We say that k is the diameter of M. The length of M is equal to the number of hexagonal strips.

Definition 28 *We use the symbol $\mathcal{N}_k$ to denote the set of carbon nanotubes of diameter k.*

We have that $\mathcal{N}_k$ is a tally set of molecules (molecular graphs). Moreover, we have that the carbon nanotubes that can be synthesized, and which are of interest in nanotechnology, are nanotubes of small diameter: the diameter is bounded above by a suitable constant N.

The set $\bigcup_{5\leq k\leq N} \mathcal{N}_k$ is a sparse set of molecular graphs, and we have to observe that we can efficiently solve an algorithmic problem about the set $\bigcup_{5\leq k\leq N} \mathcal{N}_k$, if and only if, we can solve the restriction of this problem to each one of the tally sets in the list $\mathcal{N}_5, ..., \mathcal{N}_N$. Thus, let us focus on the cases $k = 5, 6$. The cylinders in $\mathcal{N}_k$ can be capped using a carbon molecule constituted by hexagons and pentagons. Thus, any cylinder in $\mathcal{N}_k$ $(k = 5, 6)$ is the truncation of a fullerene. A planar representation of a capped $\mathcal{N}_5$-nanotube is given below.

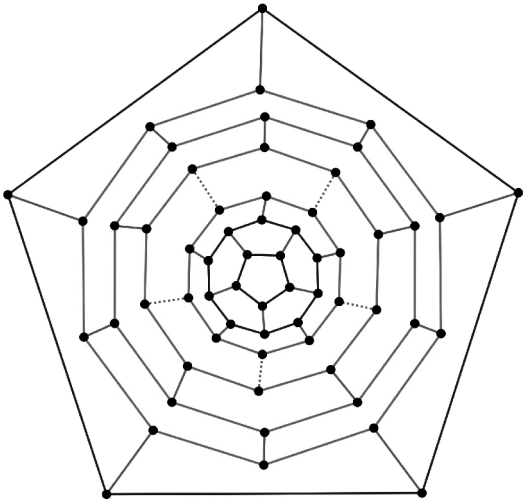

Figure 4. In this graphic we are looking into the nanotube as if it were a kaleidoscope, looking into it from one of the two pentacaps, and specifically from the pentagon that lies at the center of this pentacap.

From now on we focus on the set $\mathcal{N}_5^*$, which is constituted by all the carbon nanotubes of diameter 5 that are capped. Given $k \geq 6$, the set $\mathcal{N}_k$ can be analyzed in the same vein.

We use the symbol N_n to denote the $\mathcal{N}_5^*$-nanotube that is constituted by n hexagonal strips. From now on, and by an abuse of language, we use the term nanotube to designate the graphs in the set $\{N_n : n \geq 1\}$.

4.2 Nanotubes, automata tapes, and census functions

We can identify nanotube N_n with its planar representation, and we can think of this planar graph as a finite tape constituted by $n+2$ cells. We assume that those nested cells are scanned from the outermost to the innermost.

We can choose to work with input alphabet $\Sigma_5 \cup \Sigma_6$, where Σ_5 is constituted by all the subgraphs of a pentacap and Σ_6 is constituted by all the subsets of a hexagonal strip of diameter 5. Writing a character of this alphabet on one of the cells of N_n corresponds to choose some of the edges included in this cell (*to write a character corresponds to highlight some edges using one or more colors*).

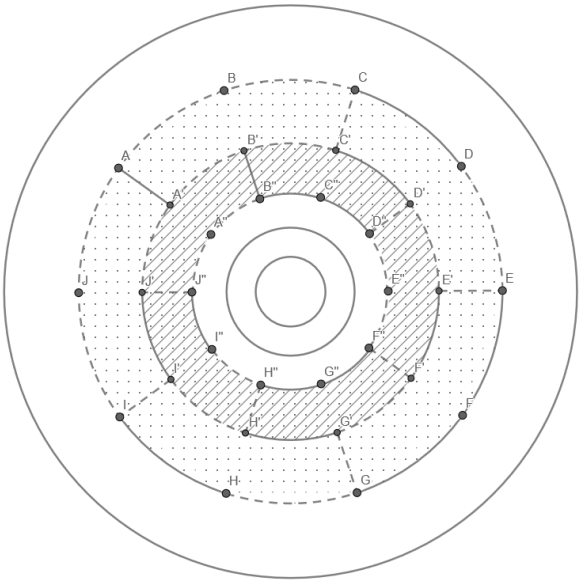

Figure 5

Suppose $i \neq 1, n+2$, and let us focus on the i-th hexagonal strip (see the Figure 5, the strip decorated with points).

We chose, in the above example, 14 edges within cell i, in this case the edges represented by broken lines. Those edges constitute a subgraph of cell i, and they also constitute a character of Σ_6. Then, we left cell i to go to cell $i+1$ (decorated with lines) and we chose some edges within cell $i+1$, which are, once again, represented by broken lines. We did the latter in a coherent way: note that cells i and $i+1$ have some common edges, the edges

$$\{A', B'\}, ..., \{I', J'\}, \{J', A'\},$$

and note that one can choose a subset of those edges when he writes a character on cell i, and then he can choose a different subset of those edges when he goes to cell $i+1$ and writes a character on this latter cell. We could, for instance, go to cell $i+2$ and choose the edge $\{I'', J''\}$. If we do this, the $(i+1)$-th character and the $(1+2)$-th character would become *inconsistent*.

Let $w \in \Sigma_5 \circ \Sigma_6^n \circ \Sigma_5$. We have that w is a sequence of cell-subgraphs that could represent (constitute) a subgraph of N_n. We have that w encodes a subgraph of G, if and only if, consecutive characters are *consistent*. Consistency is a local property that can be easily checked by finite state automata. We get that

Proposition 29 *The language*

$$SUB = \left\{w \in \Sigma_5 \circ \Sigma_6^* \circ \Sigma_5 : w \text{ represents a subgraph of } N_{|w|-2}\right\}$$

is a regular language.

Counting matchings in N_n reduces to compute the quantity

$$|\{w \in \Sigma_5 \circ \Sigma_6^n \circ \Sigma_5 : w \in SUB \text{ and } w \text{ represents a matching}\}|.$$

It is easy to check that

$$\left\{w \in \Sigma_5 \circ \Sigma_6^* \circ \Sigma_5 : w \in SUB \text{ and } w \text{ represents a matching of } N_{|w|-2}\right\}$$

is also regular language. We get that counting matchings in nanotubes reduces to compute the *census function* of an specific regular language. Is this useful?

4.3 Census functions of regular languages and Schützenberger method

Let L be a formal language. The census function of L is the tally counting problem $f_L : \mathbb{N} \to \mathbb{N}$ that is defined by the equation:

$$f_L(n) = |\{w \in \Sigma^n : w \in L\}|.$$

Census functions of regular languages are easy to compute. Let $\mathcal{M}$ be a nondeterministic finite state automaton (NFA, for short) and let $L(\mathcal{M})$ be the language accepted by $\mathcal{M}$. We have:

Theorem 30 *The census function $f_{L(\mathcal{M})}$ can be computed in time $O(\log(n))$ on the Word-Level RAM model.*

Proof. Let L be a regular language, and let $\mathcal{M}$ be a NFA accepting L. We can compute in time $O(1)$ a deterministic finite state automaton (DFA, for short) that accepts the same language as $\mathcal{M}$. Let $\mathcal{N} = (Q, q_0, A, \Sigma, \delta)$ be this DFA, and suppose that $Q = \{1, ..., q\}$. Let us suppose that q_0, the initial state, is equal to 1. The transition matrix of $\mathcal{N}$ is the matrix $T_{\mathcal{N}} = [n_{ij}]_{i,j \leq q}$, where n_{ij} is equal to

$$|\{a \in \Sigma : \delta(i, a) = j\}|,$$

that is: n_{ij} is equal to the number of characters of the input alphabet Σ that make $\mathcal{N}$ change its inner state from i to j. We can think of $\mathcal{N}$ as a multidigraph $G_{\mathcal{N}}$ whose set of nodes is equal to $\{1, ..., q\}$, and such that n_{ij} is the number of directed edges that go from i to j. We have that $T_{\mathcal{N}}$ is the adjacency matrix of $G_{\mathcal{N}}$, and we have that the number of paths of length n that go from 1 to $A \subset \{1, ..., q\}$ is equal to $\sum_{i \in A} T_{\mathcal{N}}^n[1, i]$, where $T_{\mathcal{N}}^n[1, i]$ denotes the entry $(1, i)$ of the n-th power of matrix $T_{\mathcal{N}}$. We also have that this quantity is equal to $f_{L(\mathcal{M})}(n)$. We get that the census function of $\mathcal{M}$ can be computed in time $O(n)$ using a naive algorithm for the computation of matrix powers, and we also get that this quantity can be computed in time $O(\log(n))$ over the Word-Level RAM model using fast exponentiation. The theorem is proved. ■

Remark 31 *The Word-Level RAM is a parallel model of computation that allows us to compute the n-th power of a matrix of size $O(1)$ in time $O(\log(n))$ [25].*

The above theorem provides us with a strategy that can be used to deal with many tally counting problems. We used this strategy in the previous section when we showed how to count matchings in nanotubes: we construct a bijective coding of nanotube matchings as words of a regular language, and we got a logarithmic time reduction of this problem into the problem of computing the census function of a regular language.

4.3.1 Closed formulas

Let $\mathcal{M}$ be a DFA with k states, and suppose that the transition matrix $T_{\mathcal{M}}$ can be diagonalized over the real numbers. We get that $f_{L(\mathcal{M})}$ can be expressed as a sum of k exponential functions and we get that $f_{L(\mathcal{M})}$ has a short closed formula. We also get that any tally counting problem that can be suitably reduced to $f_{L(\mathcal{M})}$ also has a closed formula. Now suppose that $T_{\mathcal{M}}$ cannot be diagonalized over the real numbers. We have [26]:

Theorem 32 *There exists $s \in \mathbb{N}$, there exist $\lambda_1, ..., \lambda_s \in \mathbb{C}$ and there exist complex polynomials $p_1(z), ..., p_s(z)$ such that for all $n \in \mathbb{N}$ the equality*

$$f_{L(\mathcal{M})}(n) = p_1(n)\lambda_1^n + \cdots + p_s(n)\lambda_s^n$$

holds. Moreover, the complex numbers $\lambda_1, ..., \lambda_s$ and the polynomials $p_1(z), ..., p_s(z)$ can be effectively computed.

Let $f : \mathbb{N} \to \mathbb{N}$ be a tally counting problem, let $\mathcal{M}$ be a DFA, and let $g, h : \mathbb{N} \to \mathbb{N}$ be two functions that hold short closed formulas. Suppose that for all n the equality $f(n) = h\left(f_{L(\mathcal{M})}(g(n))\right)$ holds. We get, as a corollary of the above theorem, that problem f admits a short closed formula. Schützenberger Method is a strategy that is used to compute closed formulas for tally counting problems. One uses this method to solve problem f when he computes a DFA $\mathcal{M}$ and a suitable reduction of f into $f_{L(\mathcal{M})}$. We can use this method to compute a closed formula for the number of matchings in nanotubes, and we can also use this method to solve many other counting problems about nanotubes. To do the latter we use the same trick we used before: we think of nanotubes as automata tapes, and we think of the substructures to be counted as strings written on those tapes that satisfy a local property which can be checked by finite state automata. Let us ask: which counting problems about nanotubes can be solved this way? We will be back with this question in the next sections.

4.4 Beyond closed formulas: Tally optimization problems

The basic insight behind Schützenberger Method is that algorithmic problems about finite automata use to be easy. We can use this idea to deal with problems about nanotubes that are of very different nature. We can deal with tally counting problems, and we can also deal with *tally optimization problems*. We can use this *Extended Schützenberger Method* to compute the Clar number of nanotubes in logarithmic time. We will not get a short closed formula in this case since, as we will see, we will loss the spectral decomposition of transition matrices.

Let $\mathcal{M}$ be a finite state automaton with input alphabet Σ, and let $f : \Sigma \to \mathbb{Q}^+$ be a weight function. We consider the following tally optimization problem about the pair $(\mathcal{M}, f)$:

Problem 33 WRL $(\mathcal{M})$, *Weightiest String Accepted by* $\mathcal{M}$.

- *Input:* n, *where* n *is a positive integer.*
- *Problem: compute the weight of the weightiest string in* $\Sigma^n \cap L(\mathcal{M})$.

We have

Theorem 34 *Problem* **WRL** $(\mathcal{M})$ *can be solved in time* $O(\log(n))$ *over the Word-Level RAM model.*

Proof. Let $(\mathcal{M}, \Sigma, f, n)$ be an instance of **WRL** $(\mathcal{M})$. Suppose

$$\mathcal{M} = (Q, q_0, A, \delta, \Sigma),$$

and suppose $Q = \{1, ..., k\}$. We use the symbol $T_{trop}(\mathcal{M})$ to denote the matrix $[t_{ij}]_{i,j \leq k}$ defined by

$$t_{ij} = \begin{cases} -\max\{f(a) : \delta(i,a) = j\}, \\ 0, \text{ otherwise.} \end{cases}$$

Let G be the weighted digraph whose adjacency matrix is equal to $T_{trop}(\mathcal{M})$, and let i, j be two states of $\mathcal{M}$. The weightiest string of length n that sends state i to state j is equal to the labeling of the lightest path of length n that connects those two vertices. Thus, it remains to look for the weight of the lightest path of length n that is directed

from the initial state of $\mathcal{M}$ to the set A, that is: if we suppose $q_0 = 1$, then it only remains to compute the quantity

$$-\min\left\{d_{\mathcal{M}}^{(n)}(1,j) : j \in A\right\}.$$

Let B, C be two $k \times k$ matrices, the *tropical product* of the pair B, C is the matrix $[d_{ij}]_{i,j \leq k}$ that is defined by

$$d_{ij} = \min\left\{b_{ir} + c_{ri} : r \leq k\right\}.$$

We use the symbol $B \otimes C$ to denote this product. We have that this tropical product is an associative operation that satisfies the following: let i, j be two nodes of G, the equality $t_{ij}^{(n)} = d_{\mathcal{M}}^{(n)}(i,j)$ holds for all n, where $t_{ij}^{(n)}$ is the ij-entry of the n-th *tropical power* of $T_{trop}(\mathcal{M})$, that is:

$$\left[t_{ij}^{(n)}\right]_{i,j \leq k} = \bigotimes_{i \leq n} T_{trop}(\mathcal{M}).$$

Notice that $T_{trop}(\mathcal{M})$ is a matrix of size $O(1)$. Then, one can compute the matrix $\bigotimes_{i \leq n} T_{trop}(\mathcal{M})$ in time $O(\log(n))$ over the Word-Level RAM model using fast exponentiation. Moreover, given the matrix $\left[t_{ij}^{(n)}\right]_{i,j \leq k}$, one can compute in time $O(1)$ the quantity $-\min\left\{t_{1i}^{(n)} : i \in A\right\}$, which is the weight of the weightiest string of length n that is included in the language $L(\mathcal{M})$. The theorem is proved. ■

Recall that we can think of the tally set $\{N_n : n \geq 1\}$ as if it were the set of finite tapes of a DFA. We can use the alphabet $\Sigma_5 \cup \Sigma_6$ to encode the subgraphs of nanotubes as strings. Moreover, we know that the string $w \in \Sigma_5 \circ \Sigma_6^n \circ \Sigma_5$ represents a subgraph of N_n, if and only if, this string belongs to the regular language SUB. We can also choose a different alphabet. If we choose the alphabet $(\Sigma_5 \times \Sigma_5) \cup (\Sigma_6 \times \Sigma_6)$, we can use this alphabet to encode pairs of substructures, as for example a perfect matching M together with an independent set of resonant hexagons for this matching. It is not hard to check that one can construct a DFA that accepts the set constituted by those pairs. Then, it only remains to define a suitable weight function

$$w_{Clar} : (\Sigma_5 \times \Sigma_5) \cup (\Sigma_6 \times \Sigma_6) \to \mathbb{N}.$$

We define this function as follows

$$w_{Clar}(\alpha, \beta) = \begin{cases} 0, \text{ if } (\alpha, \beta) \in \Sigma_5 \times \Sigma_5 \\ \text{number of hexagons included in } \beta, \text{ if } (\alpha, \beta) \in \Sigma_6 \times \Sigma_6 \end{cases}.$$

There exists a regular language

$$L_{Clar} \subset (\Sigma_5 \times \Sigma_5) \circ (\Sigma_6 \times \Sigma_6)^* \circ (\Sigma_5 \times \Sigma_5)$$

such that:

1. $w \in L_{Clar,}$ if and only if, w encodes a pair (M, H) constituted by a perfect matching of $N_{|w|-2}$ and an independent set of resonant hexagons for the matching M.

2. The Clar Number of N_n is equal to the weight of the weightiest string included in the set $L_{Clar} \cap (\Sigma_5 \circ \Sigma_6^n \circ \Sigma_5)$.

We get that the Clar number of nanotubes can be computed in logarithmic time using the tropical version of Schützenberger Method.

4.5 Beyond carbon nanotubes: Linear lattices

A nanotube of length n is built by gluing together, along a spatial axis, n copies of a pattern-graph that we called *the hexagonal strip of diameter k*. Mathematical nanosciences are specially concerned with the study of tally sets of molecular graphs that exhibit the same *one-dimensional character*. Important examples of this type of structures are linear polymers, one-dimensional crystals, carbon nanotubes of diameter k (with $k \geq 5$), boron nitride nanotubes etc. We use the term *linear lattices* to designate this type of chemical chains.

4.5.1 Linear lattices

Let (G, c) be a pair constituted by a graph G (either planar or non-planar) and a drawing of G in the plane. We suppose that the drawing c holds the following properties:

1. the drawing c fills a square of side 1.

2. The square cell is made of two vertical segments that we call L and D, together with two horizontal segments. We suppose that c places on L the same number that it places on D.

We can use this drawing of G as a pattern, and built linear chains based on this pattern. To begin with the construction we list the nodes placed on L. To do the latter

we start with the node on the top, and then we go down the segment L until we reach the node on the bottom. Let

$$w_1 = w_N, w_2, ..., w_{k-1}, w_S = w_k$$

be this ordered list. We do the same with the nodes placed on D. Let

$$e_1 = e_N, e_2, ..., e_{k-1}, e_S = e_k$$

be this second list. Now, we can glue together two copies of (G, c) and obtain a *chain* $\mathcal{C}^2(G, c)$ (a (G, c)-chain of length 2):

Let us suppose that we have two copies of (G, c) (of the chosen drawing). We use the superindex 1 to indicate that a node (or and edge) belongs to the first copy, and we use the superindex 2 to indicate that the corresponding object belongs to the second copy. We glue those two copies by placing the second copy on the right and identifying the pairs

$$\left(e_N^1, w_2^2\right), \left(e_2^1, w_2^2\right), ..., \left(e_{k-1}^1, w_{k-1}^2\right) \text{ and } \left(e_S^1, w_S^2\right).$$

The graphic below despicts two copies of a suitable planar drawing of the cubic diamond.

We use the symbol (CD, c) to denote this pattern-graph (to denote the pair constituted by the non-planar graph CD and the planar drawing c). If we identify the pair of nodes represented by hexagons, the pair of nodes represented by disks and the pair of nodes represented by triangles we get a linear-diamond of length 2. Observe that we can glue together n copies of (CD, c) and construct the chain $\mathcal{C}^n(CD, c)$. We can do the same with any pattern-graph (G, c).

Definition 35 *A linear lattice with cell-pattern (G, c) is a chain $\mathcal{C}^n(G, c)$. We say that $\mathcal{C}^n(G, c)$ is a linear lattice of length n.*

Remark 36 *Linear diamonds are more abstract objects than actual molecules. However, this abstract model of crystals can help us to understand crystal growth.*

Let us stress, once again, that there are examples of linear lattices that are of great importance in nanotechnology: carbon nanotubes, boron nitride nanotubes, linear polymers etc.

We can study some physico-chemical properties of one-dimensional molecules using chemical indices. Examples of chemical indices that are relevant are the number of

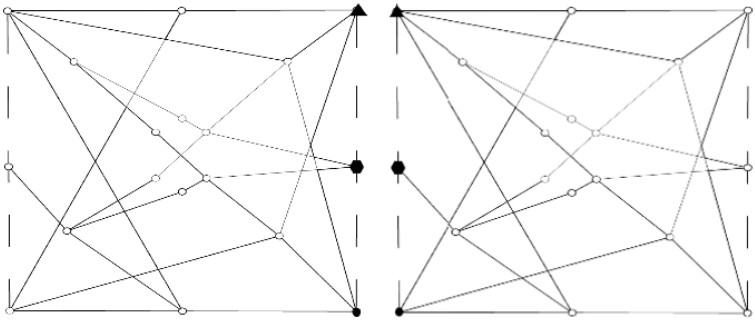

Figure 6. The broken lines indicate that the corresponding edges do not belong to the set of edges of this graph.

matchings (*Hosoya index* [27]) and the number of Hamiltonian circuits (*dense-packing number,* see [28] and [29]). If we restrict those two indices to a tally set $\{\mathcal{C}^n(G,c)\}_{n\geq 1}$, we get the tally counting problems $\#_1 Matchings_G, \#_1 Ham_G : \mathbb{N} \to \mathbb{R}$ that are defined by

$$\begin{aligned} \#_1 Matchings_G(n) &= \#\text{ matchings in the graph } \mathcal{C}^n(G,c). \\ \#_1 Ham_G(n) &= \#\text{ Hamiltonian circuits in the graph } \mathcal{C}^n(G,c). \end{aligned}$$

Note that we can use the periodic structure of the graphs in the sequence $\{\mathcal{C}^n(G,c)\}_{n\geq 1}$ to compute a suitable recurrence for the function $\#_1 Matchings_G(n)$. The counting of Hamiltonian circuits seems to be very much harder. Let us ask: can we count Hamiltonian circuits in linear lattices?

Notation 37 *Let f be a counting problem about graphs, let G be a graph, let c be a drawing of G, and let $f\restriction_{\mathcal{C}(G,c)}$ be the restriction of f to the set $\{\mathcal{C}^n(G,c)\}_{n\geq 1}$. We say that $f\restriction_{\mathcal{C}(G,c)}$ is a tally counting problem about linear lattices. We ask: what are the tally counting problems about linear lattices that can be solved in logarithmic time?*

4.6 Beyond linear time: The SBC-theorem for linear lattices

We are in the search of an algorithmic meta-theorem telling us that a large family of tally counting and tally optimization problems about linear lattices can be solved in logarithmic time. Courcelle Theorem is the paradigmatic algorithmic meta-theorem [30]. This theorem establishes that decision, counting and optimization problems that can be defined by monadic second order formulas become linear time solvable when restricted to sets of bounded treewidth [31]. We have to observe that any tally set of linear lattices is a set of bounded treewidth (bounded pathwidth). We get that any tally counting (optimization) problem about linear lattices that can be defined by a monadic second order formula can be solved in linear time. On the other hand, we have to observe that Courcelle's method does not yield algorithmic solutions that run in logarithmic time over the Word-Level RAM model. Then, if we could prove that all the problems that can be defined by monadic second order formulas can be solved in logarithmic time when restricted to tally sets of linear lattices, we would establish a new, independent and (we strongly believe) useful result. It is important to stress that monadic second order logic is an expressive logic that can be used to define many different types of graph substructures as for example Hamiltonian cycles, self-avoiding walks, matchings, perfect matchings, Eulerian orientations and Clar sets (see [32] and the references therein).

4.6.1 The SBC-theorem

The prefix SBC is an acronym for Schützenberger-Büchi-Courcelle. This theorem asserts that we can solve in time $O(\log(n))$ any problem about nanotubes and linear lattices that can be defined by a monadic second order formula. The theorem holds for counting as well as for optimization problems. We make the proof for counting problems.

From now on we think of graphs as relational structures over the vocabulary $\tau_{graph} = \{V, E, I\}$, where:

1. The unary relation V defines the set of nodes.
2. The unary relation E defines the set of edges.
3. The binary relation I corresponds to the node-edge incidence relation.

Let G be a graph, let c be a drawing of G and let $\phi(X_1, ..., X_m)$ be a monadic second order formula in the vocabulary τ_{graph}.

Consider the tally counting problem $\#_{(G,c)}\phi(X_1,...,X_m)$ that is defined by:

- *Input: n, where n is a positive integer.*
- *Problem: compute the quantity*

$$|\{H_1,...,H_m \subseteq \mathcal{C}^n(G,c) : \mathcal{C}^n(G,c) \vDash \phi(H_1,...,H_m)\}|.$$

We prove that all those tally counting about linear lattices can be solved in time $O(\log(n))$.

Theorem 38 ***The SBC-Theorem***

Problem $\#_{(G,c)}\phi(X_1,...,X_m)$ can be solved in time $O(\log(n))$.

Proof. Let us suppose that G is equal to the pair

$$(\{1,...,m\};\{e_1,...,e_r\}).$$

We use the symbols

$$b_1^i,...,b_r^i,b_{r+1}^i,...,b_{r+m}^i$$

to denote the corresponding elements of the i-th cell (bag) of $\mathcal{C}^n(G,c)$. We suppose that the last k nodes of the i-th cell are equal to the first k nodes of the $(i+1)$-cell. We also suppose that the last l edges of the i-th cell are equal to the first l edges of the $(i+1)$-th cell.

Let H be a subset of edges of $\mathcal{C}^n(G)$. We can associate to H a string (a labeled path) of length n that carries all the information needed to reconstruct this subgraph of $\mathcal{C}^n(G)$. The labeling of a node $i \in \{1,...,n\}$ must describe the intersection of H with the i-th cell of $\mathcal{C}^n(G)$. Thus we define the label $\lambda_H(i)$ as

$$\lambda_H(i) = \left\{j \leq r+m : b_j^i \text{ belongs to the intersection of the } i\text{-th cell and } H\right\}.$$

We set

$$w_H = (\{1,...,n\},\leq,\lambda_H).$$

Note that the alphabet of w_H is

$$\Sigma_G = \mathcal{P}ow(\{1,...,r+s\}).$$

We use the symbol τ_G to denote the relational vocabulary $\{R_a : a \in \Sigma_G\} \cup \{\leq\}$. To finish with the proof we have to define a suitable translation of the set of monadic second order formulas over the vocabulary τ_{graph} into the set of monadic second order formulas over the vocabulary τ_G. The latter suffices because Büchi's Theorem estates that the monadic second order properties of labeled paths can be recognized by finite automata [33]. Given $H \subseteq \mathcal{C}^n(G)$, and given $t \leq r+s$ we set

$$U_t(H) = \left\{ i \leq n : b_t^i \in H \right\},$$

and we also set $U(H) = (U_1(H), ..., U_{r+s}(H))$. Note that we are representing subsets of $\mathcal{C}^n(G)$ as $(r+s)$-tuples of subsets of $\{1, ..., n\}$. A key fact is that we can characterize the $(r+s)$-tuples of $\{1, ..., n\}$ that represent subsets of $\mathcal{C}^n(G)$. We have:

Let $U = (U_1, ..., U_{r+s})$, there exists H such that $U(H) = U$, if and only if, for all $i \in \{1, ..., n\}$ and for all $p, q, h, u \in \{1, ..., r+s\}$ we have that:

1. If h is equal to the edge $\{p, q\}$ and $i \in U_h$, then $i \in U_p, U_q$.

2. Suppose that p is one of the first k nodes of G, and suppose q is equal to $r - p + 1$, we have that $i + 1 \in U_p$, if and only if, $i \in U_q$.

3. Suppose that h is one of the first l edges of G, and suppose u is equal to $r+s-p+1$, we have that $i + 1 \in U_h$, if and only if, $i \in U_u$.

We can construct two monadic second order formulas $set(X_1, ..., X_{r+s})$ and $elem(X_1, ..., X_{r+s})$ such that for all $(r+s)$-tuples $U = (U_1, ..., U_{r+s})$ of subsets of $\{1, ..., n\}$

$$\begin{aligned} w_H &\vDash set(X_1, ..., X_{r+s}), \text{ if and only if,} \\ \text{there is } H &\subseteq \mathcal{C}^n(G) \text{ such that } U = U(H), \\ w_H &\vDash elem(X_1, ..., X_{r+s}), \text{ if and only if,} \\ \text{there is } a &\in \mathcal{C}^n(G) \text{ such that } U = U(\{a\}). \end{aligned}$$

To finish with the proof of the Theorem it suffices to prove the following claim.

Claim. *Let H be a subgraph of $C^n(G)$. We have that every monadic second order formula $\phi(X_1, ..., X_p, y_1, ..., y_q)$ in the vocabulary τ_{graph} can be effectively translated into a monadic second order formula $\phi^*\left(\widehat{X_1}, ..., \widehat{X_p}, \widehat{Y_1}, ..., \widehat{Y_q}\right)$ in the vocabulary τ_G, and such*

that for all $H_1, ..., H_p \subseteq H, a_1, ..., a_q \in H$, *it happens that*

$$\begin{aligned} H &\vDash \phi(H_1, ..., H_p, a_1, ..., a_q), \text{ if and only if,} \\ w_H &\vDash \phi^*(U(H_1), ..., U(H_p), U(\{a_1\}), ..., U(\{a_q\})). \end{aligned}$$

(*Proof of the claim*).

We begin considering the atomic case. For the formula $I(y_1, y_2)$ we set

$$(I(y_1, y_2))^* = \exists x \left(\bigvee_{i_1, i_2 \in \{1,...,r+s\}} \left(Y_{1,i_1}(x) \wedge Y_{2,i_2}(x) \wedge \bigvee_{\substack{c \in \Sigma_K \\ \{i_1, i_2\} \in c}} R_c(x) \right) \right);$$

that is, we express that there must be some path node contained in all $\widehat{Y_i}$ and such that the corresponding elements are related by I. Equalities, and formulas $V(x)$, $E(x)$ are treated similarly.

Now consider an atomic formula $\phi(X, y)$ equal to $X(y)$. We set

$$\phi^*\left(\widehat{X}, \widehat{Y}\right) = \exists x \left(\bigvee_{i \in \{1,...,r+s\}} (Y_i(x) \wedge X_i(x)) \right).$$

For the inductive step, Boolean connectives are handled in the straightforward way. To deal with quantifiers, we use the formulas *elem* and *set*. For example, if $\phi = \exists y \psi$ we let

$$\phi^* = \exists Y_1 \cdots \exists Y_{r+s} (elem(Y_1, ..., Y_{r+s}) \wedge \psi^*),$$

and if $\phi = \forall X \psi$ we let

$$\phi^* = \forall X_1 \cdots \forall X_{r+s} (set(X_1, ..., X_{r+s}) \Rightarrow \psi^*).$$

It is easy to check that the above translation works.

Let $\phi(X_1, ..., X_m)$ be a monadic second order formula over the vocabulary τ_{graph}. We get, from Büchi's Theorem, that there exists a DFA $\mathcal{M}_\phi$ such that for all n the equality

$$c_{\mathcal{M}_\phi}(n) = |\{H_1, ..., H_m \subseteq \mathcal{C}^n(G) : \mathcal{C}^n(G) \vDash \phi(H_1, ..., H_m)\}|$$

holds. We get that $\#_{(G,c)}\phi(X_1, ..., X_m)$ can be solved in time $O(\log(n))$ over the Word-Level RAM model. The theorem is proved. ■

5 Concluding remarks

Mathematical nanosciences is a research field that seems to be strongly involved with the analysis of small sets of highly regular graphs. Those small sets of graphs are small enough to become sparse, and they are also regular enough as to be efficiently enumerated, becoming tally sets of graphs. We get that many algorithmic problems coming from mathematical nanosciences are tally, and we also have that:

1. The tally classes NP_1 and $\#\mathrm{P}_1$ lack examples of natural problems that are complete for them. Mathematical nanosciences seems to be a rich source of natural problems for those classes.

2. We need a complexity class different to NP to analyze the algorithmic hardness of problems that arise from mathematical nanosciences. We claim that NP_1 and $\#\mathrm{P}_1$ are the complexity classes that are most suitable to play this role.

Acknowledgment: The author thanks Universidad Nacional de Colombia, and the financial support provided through the project Hermes 44048.

References

[1] D. Welsh, *Complexity, Knots, Colourings and Counting*, Cambridge Univ. Press, Cambridge, 1993.

[2] F. Harary (Ed.), *Graph Theory and Theoretical Physics*, Academic Press, New York, 1967.

[3] G. Kirchhoff, Über die Auflösung der Gleichungen, auf welche man bei der untersuchung der linearen verteilung galvanischer Ströme geführt wird, *Ann. Phys. Chem.* **72** (1847) 497–508.

[4] P. Kasteleyn, The statistics of dimers on a lattice I. The number of dimer arrangements on a quadratic lattice, *Physica* **27** (1961) 1209–1225.

[5] E. Lieb, Residual entropy of square ice, *Phys. Rev.* **1** (1967) 162–171.

[6] L. Valiant, The complexity of computing the permanent, *Theor. Comput. Sci.* **8** (1979) 189–201.

[7] S. Toda, M. Ogiwara, Counting classes are at least as hard as the polynomial-time hierarchy, *SIAM J. Comput.* **21** (1992) 316–328.

[8] M. Luby, D. Randall, A. Sinclair, Markov chain algorithms for planar lattice structures, *SIAM J. Comput.* **31** (2001) 167–192.

[9] S. Mahaney, Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis, *J. Comput. Syst. Sci.* **25** (1982) 130–143.

[10] J. Hartmanis, N. Immerman, V. Sewelson, Sparse sets in NP-P: EXPTIME versus NEXPTIME, *Inf. Control* **65** (1985) 158–181.

[11] L. Valiant, The Complexity of enumeration and reliability problems, *SIAM J. Comput.* **8** (1979) 410–421.

[12] S. Rudich, *Lecture Notes on Computational Complexity Theory*, `https://web.cs.ucdavis.edu/~rogaway/classes/220/winter06/rudich.pdf`

[13] A. Itai, C. Papadimitriou, J. Szwarcfiter, Hamilton paths in grid graphs, *SIAM J. Comput.* **11** (1982) 676–686.

[14] M. Liskiewicz, M. Ogihara, S. Toda, The complexity of counting self-avoiding walks in subgraphs of two-dimensional grids and hypercubes, *Theor. Comput. Sci.* **304** (2003) 129–156.

[15] M. Jerrum, Two-dimensional monomer-dimer systems are computationally intractable, *J. Stat. Phys.* **48** (1987) 121–134.

[16] P. Fowler, D. Manolopoulos, *An Atlas of Fullerenes*, Dover, New York, 2006.

[17] Y. Dong, Y. Liu, H. Zhang, A combination of Clar number and Kekulé count as an indicator of relative stability of fullerene isomers of C_{60}, *J. Math. Chem.* **48** (2010) 733–740.

[18] E. Clar, *The Aromatic Sextet*, Wiley, London, 1972.

[19] N. Trinajstic, *Chemical Graph Theory*, Routledge, London, 1992.

[20] E. Berczi-Kovacs, A. Bernath, The complexity of the Clar number problem and an exact algorithm, *J. Math. Chem.* **56** (2018) 597–605.

[21] H. Abeledo, G. Atkinson, Unimodularity of the Clar number problem, *Lin. Algebra App.* **420** (2007) 441–448.

[22] C. Greenhill, The complexity of counting colourings and independent sets in sparse graphs and hypergraphs, *Comput. Complex.* **9** (2000) 52–72.

[23] P. Engel, P. Smillie, The number of non-negative curvature triangulations of S^2, arXiv:1702.02614.

[24] I. Sumio, Helical microtubules of graphitic carbon, *Nature* **354** (1993) 56–58.

[25] J. Hartmanis, J. Simon, On the power of multiplciation in random-access machines, *Proc. 15th Annu. IEEE Sympos. Switching Automata Theory* (1974) 13–23.

[26] P. Flajolet, R. Sedgewick, *Analytic Combinatorics*, Cambridge Univ. Press, Cambridge, 2009.

[27] H. Hosoya, K. Kawasaki, K. Mizutani, Topological index and thermodynamic properties I: Empirical rules on the boiling points of saturated hydrocarbons, *Bull. Chem. Soc. Jpn.* **45** (1972) 3415–3421.

[28] O. Bodroža, B. Pantić, I. Pantić, M. Bodroža-Solarov, Enumeration of Hamiltonian cycles in some grid graphs, *MATCH Commun. Math. Comput. Chem.* **70** (2013) 181–204.

[29] J. Montoya, A note concerning the algorithmic analysis of polymer thermodynamics, *MATCH Commun. Math. Comput. Chem.* **67** (2012) 761–772.

[30] B. Courcelle, The monadic second-order logic of graphs. I. Recognizable sets of finite graphs, *Inf. Comput.* **85** (1990) 12–75.

[31] B. Courcelle, J. Makowsky, U. Rotics, On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic, *Discr. Appl. Math.* **108** (2001) 23–52.

[32] B. Courcelle, J. Engelfriet, *Graph Structure and Monadic Second-Order Logic: A Language-Theoretic Approach*, Cambridge Univ. Press, Cambridge, 2012.

[33] J. Büchi, On a decision method in restricted second order arithmetic, *Proc. Int. Congr. Logic, Method, and Philosophy Sci.* (1960) 425–435.