Mixed Integer Programming and Extremal Chemical Graphs

PIERRE HANSEN
GERAD and HEC Montréal
3000, Côte-Sainte-Catherine Rd Montreal (Quebec) H3T 2A7
CANADA
pierre.hansen@gerad.ca

MUSTAPHA AOUCHICHE
GERAD and HEC Montreal
CANADA
mustapha.aouchiche@gerad.ca

GILLES CAPOROSSI
GERAD and HEC Montreal
CANADA
gilles.caporossi@gerad.ca

ALAIN HERTZ
GERAD and Polytechnique Montreal
CANADA
alain.hertz@gerad.ca

CHERIF SELLAL
GERAD and Polytechnique Montreal
CANADA
cherif.sellal@gerad.ca

Abstract: Two systems called AutoGraphiX and ChemoGraphiX are proposed for datamining chemical graphs with extremal values of one or several graphical invariants. AutoGraphiX is based on the variable neighborhood search heuristic and ChemoGraphiX on mixed integer programming.

Key–Words: Chemical graphs; Mixed integer programming; Metaheuristic; Graphical invariant

1 Introduction

Graphs $G = (V, E)$ are often used to describe molecules. The set V of vertices is associated with the atoms of the molecule, and the set $E = \{v_i v_j\}$ of edges to bonds between them. Moreover, a large number of graphical invariants, *i.e.*, formulae associated with the graphs which do not depend on the numbering of the vertices and/or edges, are used to describe the properties of those graphs. Extremal values, *i.e.*, minima or maxima, of the invariants often correspond to graphical properties or structures in QSAR and QSPR studies.

Finding algorithms for determination of these extremal graphs has been the subject of systematic studies on extremal chemical graphs, done at GERAD, Montreal during the last decade. It focusses on two different families of methods: (i) the AutoGraphiX (AGX) system which applies the variable neighborhood search metaheuristic to study properties of various families of chemical graphs, such as trees, unicyclic graphs, or trianle free graphs; (ii) the ChemoGraphiX system (CGX) which exploits the fact that such graphs have maximum degree at most 4. One can then apply mixed integer programming with variables associated with the number n_i of vertices of degree i and with the number m_{ij} of edges with end-vertices with degrees i and j. This leads to mathematical programs in 13 variables which ar easy to solve.

2 Variable neighborhood search

Metaheuristics are general frameworks to build heuristics for solving combinatorial and global optimization problems. They have been the subject of intensive research since genetic search and Simulated Annealing were proposed [21] as general schemes for building heuristics which get out of local minima or maxima. Several other metaheuristics were soon proposed. For discussion of the best-known of them the reader is referred to the books of surveys [5, 12, 25]. Some of the many successful applications of metaheuristics are also mentioned there.

Variable Neighborhood Search (VNS) [16, 17, 18, 19, 23] is a metaheuristic which exploits systematically the idea of neighborhood change, both in descent to local minima and in escape from the valleys which contain them. VNS exploits the following three observations:

• A local minimum with respect to one neighborhood structure is not necessary so for another.

• A global minimum is a local minimum with respect to all possible neighborhood structures.

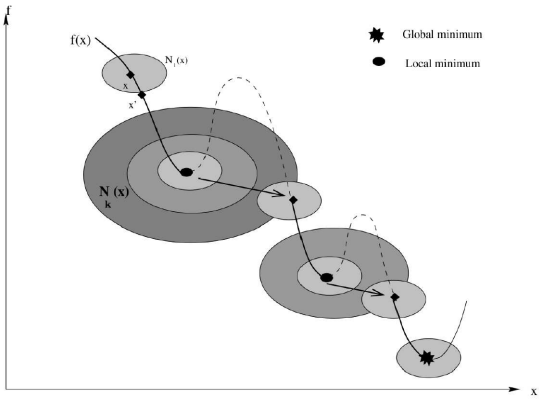

Figure 1: VNS curves

• For many problems local minima with respect to one or several neighborhoods are relatively close to each other.

Unlike many other metaheuristics, the basic schemes of VNS and its extensions are simple and require few, and sometimes no parameters. Therefore, in addition to providing very good solutions, often in simpler ways than other methods, VNS gives insight into the reasons for such a performance, which, in turn, can lead to more efficient and sophisticated implementations.

Function VNS (x, k_{max}, t_{max});
1 **repeat**
2 $k \leftarrow 1$;
3 **repeat**
4 $x' \leftarrow$ Shake (x, k);
5 $x'' \leftarrow$ FirstImprovement (x');
6 NbhoodChange (x, x'', k);
until $k = k_{max}$;
7 $t \leftarrow$ CpuTime ();
until $t > t_{max}$;

Algorithm 1: Steps of the basic VNS

The Basic VNS (BVNS) method [23] combines deterministic and stochastic changes of neighbourhood. Its steps are given in Algorithm 1 (see also Figure 1). Often successive neighbourhoods will be nested. Observe that point x' is generated at random in Step 4 in order to avoid cycling, which might occur if deterministic rules were applied. In Step 5, several neighborhoods may be used. In this case, we speak about *variable neighborhood descent* (VND), the scheme of which is given in Algorithm 2. For more details about VNS and its applications in solving problems in different domains of sciences see the recent survey [20] as well as the references therein.

In all its applications, VNS is used as an optimization tool. These applications are mainly solving specific optimization problems. However, VNS can also be used in *discovery science*, *i.e.*, help in the development of theories. The first domain to be addressed in this way was graph theory.

Function VND (x, k'_{max});
1 **repeat**
2 $k \leftarrow 1$;
3 **repeat**
4 $x' \leftarrow arg\min_{y \in \mathcal{N}'_k(x)} f(x)$;
5 NbhoodChange (x, x', k);
until $k = k'_{max}$;
until *no improvement is obtained*;

Algorithm 2: Steps of the basic VND

2.1 VNS for extremal graphs

VNS is the fundamental tool exploited in the system AutoGraphiX (AGX, for short) [2, 8, 9], which is devoted to conjecture–making, and therefore to scientific discovery, in graph theory. A long series of papers (see the list in [4]) with the common title "*Variable neighborhood search for extremal graphs*" was published. Several of the papers which use AGX without being included within this series are listed in [4]. This system addresses the following problems:
• Find a graph satisfying given constraints;
• Find optimal or near optimal graphs for an invariant subject to constraints;
• Refute a conjecture;
• Suggest a conjecture (or repair or sharpen one);
• Provide a proof (in simple cases) or suggest an idea of proof.

A basic idea is then to consider all of these problems as parametric combinatorial optimization problems on the infinite set of all graphs (or in practice some smaller subset) solved with a generic heuristic. This is done by applying VNS to find extremal graphs, with a given number n of vertices (and possibly also a given number of edges). Then a VND with many neighbourhoods is used. Those neighborhoods are defined by modifications of the graphs such as the removal or addition of an edge, rotation of an edge, and so forth. Once a set of extremal graphs, parametrized by their order, is found, their properties are explored with various data mining techniques, leading to conjectures, refutations and simple proofs or ideas of proof.

2.2 The AutoGraphiX system

Among the first application of VNS, a computer program, called the *AutoGraphiX system* (AGX, for short) [2, 8, 9], was built for conjecture–making in

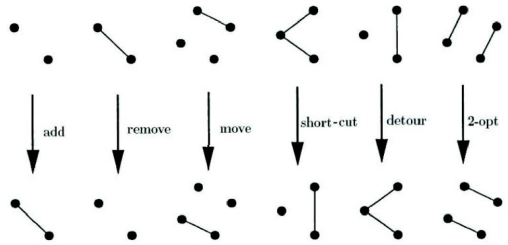

Figure 2: Examples of moves (changes) used in AGX

graph theory. This system has been developed at GERAD, Montreal, since 1997. Conjectures obtained with AGX were proved by the present authors or by graph theorists from several countries.

Graph theory is replete with theorems involving graph invariants. They are either *algebraic*, *i.e.,* equalities or inequalities involving one or several invariants, or *structural*, *i.e.,* characterizations of the families of graphs for which an invariant takes an extremal value. Both types of results can be conjectured by AGX, in a fully automated way, or in some cases, to be carefully distinguished, in an assisted way.

Let $\mathcal{G}_n$ and $\mathcal{G}_{n,m}$ denote respectively the sets of all graphs with n vertices, and with n vertices and m edges. Two basic ideas underlie the systems AGX:

• *Most problems of extremal graph theory can be viewed as problems of parametric combinatorial optimization of the form*

$$\min / \max_{G \in \mathcal{G}_n} i(G) \quad \text{or} \quad \min / \max_{G \in \mathcal{G}_{n,m}} i(G) \qquad (1)$$

for some invariant $i(G)$ with parameters n and m, or the exploitation of their solutions (in practice only moderate values of n and m will be considered);

• *All problems of the form (1) can be solved approximately by a generic heuristic.*

To obtain such a heuristic, the Variable Neighborhood Search metaheuristic (VNS) is specialized. VNS exploits systematically changes in neighborhoods used in the search (see Figure 2 for examples of changes), both in a descent phase to obtain a locally extremal graph, and in a "shaking" phase, to get out of the corresponding valley (or away from the corresponding mountain) in order to find a better graph.

Rules of VNS applied in AGX are the following:

1. Select the set of neighborhood structures N_k, $k = 1, \ldots k_{max}$ that will be used in the search for a better locally optimal graph, and a stopping condition. Choose an initial graph G.

   Repeat until the stopping condition is met:
2. Set $k = 1$;
3. Until $k = k_{max}$, repeat the following steps:
   (a) (*shaking*) generate a graph G' from the k^{th} neighborhood of G ($G' \in N_k(G)$);
   (b) (*descent*) apply VND with G' as initial graph; denote with G'' the locally optimal graph obtained;
   (c) (*improvement or continuation*) if $i(G'')$ is better than $i(G)$, the best value of i for a previously visited graph, move there, *i.e.,* replace G by G'', and continue search within $N_1(G)$; otherwise, set $k \leftarrow k + 1$.

The stopping condition is usually a maximum computing time. The optimization routine of VNS is called *variable neighborhood descent*. It explores systematically larger and larger neighborhoods of the current graph, and performs a move whenever it is profitable (first improvement) or is also best within its neighborhood (best improvement). The neighborhoods used initially in AGX are the following: remove, add, move, detour, short cut, 2–opt, insert pending vertex, add pending vertex, and remove vertex.

In the most recent version of AGX, the VND routine is replaced by *Learning Descent* (LD), in order to keep track of which transformations are the most fruitful and to reinforce their use. The LD used in AGX is described in [7].

Once a set of (presumably) extremal graphs has been found, conjectures can be stated by one of the following 3 approaches [8]:

(i) a *numerical method* which applies the mathematics of Principle Component Analysis [10] to determine, in polynomial time, a basis of affine relations between invariants, satisfied by the extremal graphs found.

(ii) a *geometric method* which views extremal graphs as points in invariants space and applies a "gift-wrapping" algorithm to find their convex hull and linear inequality relations associated with its facets. Note that a similar approach is used in GraPHedron [11];

(iii) an *algebraic method* [1, 3, 2] which recognizes to which family (or families) of graphs the extremal graphs belong, then uses a database of formulae for invariants in function of the order of G to obtain conjectures.

3 The ChemoGraphiX system

3.1 Edge realizability of simple graphs

Let $G = (V, E)$ be a graph with vertex set V and edge set E. We denote by $G[W]$ the subgraph of G induced by a subset $W \subseteq V$ of vertices, by $d_G(v)$ the degree of v in G, and by $\Delta(G)$ the maximum degree in G. Let e_{uv} be the number of edges linking u with v in G. The number $\mu(G)$ of multiple edges in G is defined as

$\mu(G) = \sum_{u \neq v} \max\{e_{uv} - 1, 0\}$. Hence, $\mu(G) = 0$ if and only if G does not contain any multiple edge.

Given a symmetric $r \times r$ matrix $M = [m_{ij}]$, an M-graph is a graph G with $\Delta(G) = r$ and such that the number of edges with end-vertex degrees i and j is equal to m_{ij}. Multiple edges contribute by their multiplicity to both of their end-degrees and loops contribute by 2 to the degree of their unique end-vertex.

Let Γ_M be the set of simple M-graphs (i.e., the set of M-graph without loops or multiple edges). The symmetric matrices M for which Γ_M is non-empty were characterized by [15] in the following theorem.

Theorem 1 ([15]) *Let $M = [m_{ij}]$ be a symmetric $r \times r$ matrix of non-negative integers. $\Gamma_M \neq \emptyset$ if and only if the following conditions hold:*

(C1) $n_i = \frac{1}{i}(\sum_{j=i}^{r} m_{ij} + \sum_{j=1}^{i} m_{ji})$ *is an integer for all* $i = 1, \ldots, r$;

(C2) $m_{ii} \leq \frac{1}{2}(n_i(n_i - 1))$ *for all* $i = 2, \ldots, r$ *such that* $1 \leq n_i \leq i$.

(C3) $m_{ij} \leq n_i n_j$ *for all* $2 \leq i < j \leq r$ *such that* $1 \leq n_i < j$ *and* $1 \leq n_j < i$.

An algorithm for the construction of a simple M-graph, based on the above theorem, is provided in [15]. More precisely, given a matrix M that satisfies conditions (C1)–(C3), the algorithm constructs a simple M-graph in three steps: (a) build an M-graph; (b) remove loops; and (c) remove multiple edges.

3.2 Edge realizability of connected simple graphs

Let $\Gamma'_M \subseteq \Gamma_M$ be the set of simple M-graphs with minimum number of connected components. The next theorem gives necessary and sufficient conditions on M so that all graphs in Γ'_M are connected. In comparison with Theorem 1, a fourth condition has to be added. Before the statement of the result, we first need to introduce some notations.

For a graph G, let $D(G)$ be the set of integers i such that there is at least one vertex v with $d_G(v) = i$ that lies on a cycle in G. Also, let $D'(G)$ be the set of integers $i \notin D(G)$ such that there is at least one vertex w with $d_G(w) = i$ that lies on a path P in G whose endpoints u and v have the same degree $d_G(u) = d_G(v) \in D(G)$. Finally, let $H(G)$ be the subgraph of G induced by the vertices with degree $i \in D(G) \cup D'(G)$ in G. For example, considering the graphs in Figure 3, we have $D(G) = \{3, 4\}$ in (a), (b), (c), $D(G) = \emptyset$ in (d), $D'(G) = \{2\}$ in (a), $D'(G) = \{2, 6\}$ in (b), and $D'(G) = \emptyset$ in (c) and (d). The black

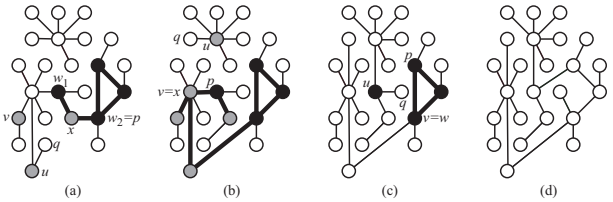

Figure 3: Example of graphs with $D(G) = \{3, 4\}$.

vertices are those with a degree $i \in D(G)$, while the grey ones are those with a degree $i \in D'(G)$. The vertices of $H(G)$ are the black and grey ones, and the edges of $H(G)$ are those represented with bold lines.

Let $\mathcal{P}_r$ be the set containing all partitions of all subsets of $\{2, \ldots, r\}$. For example, for $r = 4$, $\mathcal{P}_4$ contains the 15 following partitions:

- the 5 non-empty partitions of $\{2, 3, 4\}$: $\big\{\{2\}, \{3\}, \{4\}\big\}$,$\big\{\{2\}, \{3, 4\}\big\}$, $\big\{\{3\}, \{2, 4\}\big\}$, $\big\{\{4\}, \{2, 3\}\big\}$, and $\big\{\{2, 3, 4\}\big\}$;
- the 2 non-empty partitions of $\{2, 3\}$: $\big\{\{2\}, \{3\}\big\}$,$\big\{\{2, 3\}\big\}$;
- the 2 non-empty partitions of $\{2, 4\}$: $\big\{\{2\}, \{4\}\big\}$,$\big\{\{2, 4\}\big\}$;
- the 2 non-empty partitions of $\{3, 4\}$: $\big\{\{3\}, \{4\}\big\}$,$\big\{\{3, 4\}\big\}$;
- the 3 non-empty partitions of $\{2\}$, $\{3\}$ and $\{4\}$: $\big\{\{2\}\big\}$,$\big\{\{3\}\big\}$,$\big\{\{4\}\big\}$;
- the empty partition.

Also, for a partition $p \in \mathcal{P}_r$, let $E_r(p)$ be the set of all integers that appear in a subset of p (i.e., $E_r(p) = \cup_{s \in p}\ s$, and let $\overline{E_r(p)} = \{2, \ldots, r\} \setminus E_r(p)$. For example, for $p = \{\{2, 3\}, \{5\}\}$, we have $E_6(p) = \{2, 3, 5\}$ and $\overline{E_6(p)} = \{4, 6\}$.

Now, let $I(M)$ be the set of integers i in $\{2, \ldots, r\}$ such that $m_{ii} + \sum_{j=1}^{r} m_{ij} \geq 1$. For a partition $p \in \mathcal{P}_r$, we denote by $|p|_M$ be the number of subsets $s \in p$ such that $s \cap I(M) \neq \emptyset$. For example, for $I(M) = \{2, 3, 5, 6, 8\}$ and $p = \{\{2, 4\}, \{3\}, \{5, 8\}, \{7\}\}$, we have $|p|_M = 3$.

There is a bijection between $\mathcal{P}_r$ and the set of partitions of $\{1, \ldots, r\}$. Indeed, to every partition $p \in \mathcal{P}_r$, we can associate a partition of the set $\{1, \ldots, r\}$ by adding the bloc $\overline{E_r(p)} \cup \{1\}$ to p. Hence, the total number of partitions in $\mathcal{P}_r$ is the r^{th} *Bell number* B_r (sequence A000110 in OEIS [26]).

We are now ready for the statement of the main theorem, proved in [15], that characterizes those matrices M for which there is a simple connected M-graph G.

Theorem 2 ([15]) *Let $M = [m_{ij}]$ be a symmetric $r \times r$ matrix of non-negative integers. There is a simple connneted M-graph G if and only if the following conditions hold:*

(C1) $n_i = \frac{1}{i}(\sum_{j=i}^{r} m_{ij} + \sum_{j=1}^{i} m_{ji})$ *is an integer for all* $i = 1, \ldots, r$;

(C2) $m_{ii} \leq \frac{1}{2}(n_i(n_i - 1))$ *for all* $i = 2, \ldots, r$ *such that* $1 \leq n_i \leq i$.

(C3) $m_{ij} \leq n_i n_j$ *for all* $2 \leq i < j \leq r$ *such that* $1 \leq n_i < j$ *and* $1 \leq n_j < i$.

(C4) $\sum_{\substack{s \neq s' \\ s,s' \in p}} \sum_{\substack{i \in s \\ j \in s'}} m_{ij} + \sum_{\substack{i \in E_r(p) \\ j \in \overline{E_r(p)}}} m_{ij} + \sum_{\substack{i \leq j \\ \{i,j\} \subseteq \overline{E_r(p)}}} m_{ij} - m_{11} \geq \sum_{i \in \overline{E_r(p)}} n_i + |p|_M - 1 \; \forall p \in \mathcal{P}_r$.

While conditions (C4) of Theorem 2 are numerous, particularly for large values of r, they may prove to be useful in the case of chemical graphs, where $r = \Delta(G) \leq 4$. Indeed, $\mathcal{P}_4$ contains only 15 partitions.

Given a matrix M that satisfies conditions (C1)–(C4) of the above theorem, an algorithm for the construction of a simple connected M-graph is provided in [15].

3.3 An integer programming model

Let n and m be two positive integers. In this section, we show how to determine a symmetric $r \times r$ matrix $M = [m_{ij}]$ of non-negative integers that satisfies all conditions of Theorem 2 as well as the two following conditions:

(C5) $n = \sum_{i=1}^{r} n_i$

(C6) $m = \sum_{1 \leq i \leq j \leq r} m_{ij}$.

An M-graph with such a matrix M has n vertices and m edges. Finding such a matrix can be done using an Integer Linear Programming (ILP) model. Since M has to be symmetric, we consider non-negative integer variables m_{ij} for all $1 \leq i \leq j \leq r$. The ILP also uses non-negative integer variables n_i $(i = 1, \ldots, r)$ which are constrained as follows, to satisfy condition (C1) :

$$\sum_{j=i}^{r} m_{ij} + \sum_{j=1}^{i} m_{ji} = i n_i \;\; \forall i = 1, \ldots, r \qquad (2)$$

In order to impose condition (C2), we consider new Boolean variables x_{ik} defined for $i = 1, \ldots, r$ and $k = 1, \ldots, i$, and impose

$$n_i \geq (k+1)(1 - x_{ik}) \qquad (3)$$
$$\forall i = 2, \ldots, r, \forall k = 1, \ldots, i$$
$$m_{ii} + x_{ik} m \leq \frac{k(k-1)}{2} + m \qquad (4)$$
$$\forall i = 2, \ldots, r, \forall k = 1, \ldots, i$$

Constraints (3) imply that $x_{ik} = 1$ when $n_i \leq k$, while x_{ik} can take value 0 or 1 otherwise. Consider any $i \in \{2, \ldots, r\}$:

- if $n_i > i$, constraints (4) do not impose any restriction since x_{ik} can be set equal to 0 for all $k = 1, \ldots, i$;
- if $n_i = 0$, constraints (4) impose a series of upper bounds on m_{ii}, the strongest one being obtained with $k = 1$. We thus get $m_{ii} \leq 0$, which is already imposed by constraints (2);
- if $1 \leq n_i \leq i$, constraints (4) impose a series of upper bounds on m_{ii}, the strongest one being obtained with $k = n_i$ (i.e., $m_{ii} \leq \frac{1}{2} n_i(n_i - 1)$), which corresponds to condition (C2).

Condition (C3) is imposed in a similar way:

$$m_{ij} + m x_{jk} \leq k n_i + m$$
$$\forall 2 \leq i < j \leq r, \forall k = 1, \ldots, i-1 \qquad (5)$$
$$m_{ij} + m x_{ik} \leq k n_j + m$$
$$\forall 2 \leq i < j \leq r, \forall k = 1, \ldots, j-1 \qquad (6)$$

Indeed, consider any i, j such that $2 \leq i < j \leq r$:

- if $n_i \geq j$ and $n_j \geq i$, no constraint is imposed since x_{jk} in (5) and x_{ik} in (6) can be set equal to 0 for all considered values of k;
- if $n_i = 0$ or $n_j = 0$, constraints (5) and (6) are not more restrictive than constraints (2) which impose $m_{ij} = 0$;
- if $n_i \geq j$ and $1 \leq n_j \leq i - 1$, constraints (5) impose $m_{ij} \leq n_i n_j$, which is not more restrictive than $m_{ij} \leq j n_j$ imposed by constraints (2) (while constraints (6) do not impose any restriction);
- if $n_j \geq i$ and $1 \leq n_i \leq j - 1$, constraints (6) impose $m_{ij} \leq n_i n_j$, which is not more restrictive than $m_{ij} \leq i n_i$ imposed by constraints (2).
- if $1 \leq n_j \leq i - 1$ and $1 \leq n_i \leq j - 1$, both (5) and (6) impose condition (C3).

For imposing condition (C4), the only difficulty is the term $|p|_M$ since M is not known. By definition, $I(M)$ is the set of integers i in $\{2,\ldots,r\}$ such that $m_{ii} + \sum_{j=1}^{r} m_{ij} \geq 1$, and it follows from constraints (2) that this is equivalent to say that $I(M)$ is the set of integers i in $\{2,\ldots,r\}$ such that $n_i \geq 1$. Hence, given a partition $p \in \mathcal{P}_r$ and a set $s \in p$, we have $s \cap I(M) \neq \emptyset$ if and only if there exists $i \in s$ with $n_i \geq 1$. We therefore define Boolean variables q_s for all non-empty subsets s of $\{2,\ldots,r\}$ so that

$$q_s = \begin{cases} 1 & \text{if there exists } i \in s \text{ with } n_i \geq 1 \\ 0 & \text{otherwise.} \end{cases}$$

This is done by imposing the following constraints:

$$\sum_{i \in s} n_i \leq n q_s \quad \forall \text{ non-empty } s \subseteq \{2,\ldots,r\} \quad (7)$$

$$\sum_{i \in s} n_i \geq q_s \quad \forall \text{ non-empty } s \subseteq \{2,\ldots,r\} \quad (8)$$

Since $|p|_M = \sum_{s \in p} q_s$, we can now impose Condition (C4) as follows. For a partition $p \in \mathcal{P}_r$, let $A(p)$ be the set of pairs (i,j) such that $i < j$ and there are two distinct sets s, s' in p with $i \in s$ and $j \in s'$. Also, let $B(p)$ the the set of pairs (i,j) such that $i < j$ and either $i \in E_r(p)$ and $j \in \overline{E_r(p)}$, or $j \in E_r(p)$ and $i \in \overline{E_r(p)}$. Condition (C4) is then imposed by the following constraint:

$$\sum_{(i,j) \in A(p) \cup B(p)} m_{ij} + \sum_{\substack{i \leq j \\ \{i,j\} \subseteq \overline{E_r(p)}}} m_{ij} - m_{11} \geq \sum_{i \in \overline{E_r(p)}} n_i + \sum_{s \in p} q_s - 1 \quad \forall p \in \mathcal{P}_r \quad (9)$$

Clearly, conditions (C5) and (C6) are imposed as follows, where n and m are fixed integers.

$$\sum_{i=1}^{r} n_i = n \quad (10)$$

$$\sum_{1 \leq i \leq j \leq r} m_{ij} = m \quad (11)$$

Finally, the following constraints define the possible values of all variables:

$$m_{ij} \in \mathbb{N} \quad \forall i = 1,\ldots,r, \forall j = i,\ldots,r \quad (12)$$

$$n_i \in \mathbb{N} \quad \forall i = 1,\ldots,r \quad (13)$$

$$x_{ik} \in \{0,1\} \quad \forall i = 2,\ldots,r, \forall k = 1,\ldots,i \quad (14)$$

$$q_s \in \{0,1\} \quad \forall \text{ non-empty } s \subseteq \{2,\ldots,r\} \quad (15)$$

A simple calculation shows that there are $2^{r-1} + r(r+2) - 2$ variables and $2^r + \frac{r}{2}(r^2 - r + 6) - 4 + B_r$ constraints (where B_r denotes the r^{th} Bell number).

3.4 Finding more than one matrix

Given any matrix M produced by the ILP of the previous section, we now show how to generate a different one (if any) that also satisfies conditions (C1)–(C6). This is done as follows. Let $\{M_{ij}\}$ denote the values of the matrix obtained using the ILP of the previous section. For all $1 \leq i \leq j \leq r$ with $M_{ij} > 0$, we define a Boolean variable y_{ij} so that $y_{ij} = 1$ if and only if $M_{ij} < m_{ij}$. This is done by imposing the following constraints:

$$M_{ij} + m y_{ij} \geq m_{ij} \quad \forall 1 \leq i \leq j \leq r \text{ with } M_{ij} > 0 \quad (16)$$

$$y_{ij}(M_{ij} + 1) \leq m_{ij} \quad \forall 1 \leq i \leq j \leq r \text{ with } M_{ij} > 0 \quad (17)$$

$$y_{ij} \in \{0,1\} \quad \forall 1 \leq i \leq j \leq r \text{ with } M_{ij} > 0 \quad (18)$$

In a similar way, we consider, we consider Boolean variable z_{ij} so that $z_{ij} = 1$ if and only if $M_{ij} > m_{ij}$:

$$M_{ij} + m(1 - z_{ij}) \geq m_{ij} + 1 \quad \forall 1 \leq i \leq j \leq r \text{ with } M_{ij} > 0 \quad (19)$$

$$(1 - z_{ij}) M_{ij} \leq m_{ij} \quad \forall 1 \leq i \leq j \leq r \text{ with } M_{ij} > 0 \quad (20)$$

$$z_{ij} \in \{0,1\} \quad \forall 1 \leq i \leq j \leq r \text{ with } M_{ij} > 0 \quad (21)$$

In order to generate a new matrix different from the previous one, it is then sufficient to add the constraint that at least one the y_{ij} and z_{ij} variables must be equal to 1. This is simply done as follows:

$$\sum_{i=1}^{r} \sum_{j=i}^{r} (y_{ij} + z_{ij}) \geq 1 \quad (22)$$

3.5 Extremal graphs for some Adriatic indices

For a graph G, let $\mathcal{I}(G)$ be an invariant that can be written as a function linear in the numbers n_i of vertices of degree i and in the numbers m_{ij} of edges with end-degrees i and j. For example the first and second Zagreb indices [13] and the Randić index [24] are

defined as follows :

$$\text{First Zagreb index of } G \quad : \quad \sum_{i=1}^{r} n_i i^2$$

$$\text{Second Zagreb index of } G \quad : \quad \sum_{i=1}^{r} \sum_{j=i}^{r} m_{ij} ij$$

$$\text{Randić index of } G \quad : \quad \sum_{i=1}^{r} \sum_{j=i}^{r} \frac{m_{ij}}{\sqrt{ij}}$$

Such indices belong to the set of Adriatic indices studied in [27]. In this section, we show how to determine simple graphs and simple connected graphs that optimize (i.e., minimize of maximize) these invariants.

Given two integers n and m, finding a simple connected graph G with optimal value $\mathcal{I}(G)$ can be done by solving the following ILP, and then building a simple connected M-graph (with Algorithm 2), using the matrix M produced by the ILP:

Minimize or maximize the graph invariant $\mathcal{I}$
Subject to constraints (1)–(14)

We illustrate the use of the models by considering *chemical trees*, i.e., trees with maximum degree $r \leq 4$. We therefore solve the ILP by setting $r = 4$ and $m = n - 1$. As already mentioned in Section 3.3, the ILP has $2^{r-1} + r(r+2) - 2$ variables and $2^r + \frac{r}{2}(r^2 - r + 6) - 4 + B_r$ constraints, which gives a total of 30 variables and 63 constraints for $r = 4$, regardless of the number of vertices in the considered chemical trees. We first identify all simple chemical trees with $6 \leq n \leq 15$ vertices having minimum, second-minimum, third-ninimum, fourth minimum, and fifth-minimum value of the Randić index. The set of extremal chemical trees is shown in Figure 5.

For comparison, a similar study was performed in [14] and [22], where the authors analyse chemical trees with minimum, second minimum and third minimum Randić index. They give one example of such extreme graphs for every $n = 6, 7, \ldots, 24$. A careful comparison of the these studies shows that three graphs presented in [14] and [22] (at page 87) are not correct: their second-minimum and third-minimum for $n = 11$, and their third minimum for $n = 14$ have a Randić index strictly larger than our fifth-minimum. For example, the graphs shown in Figure 5 with $n = 11$ have a Randić index of 4.5, 4.62, 4.65, 4.66, and 4.69, while the graph presented in [14] and [22] as second-minimum, and drawn in Figure 4, has value 4.71.

Condition (C4) is essential to ensure the connectivity. For comparison, we show in Figure 6 the simple graphs having maximum degree $r \leq 4$, $m = n - 1$

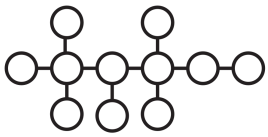

Figure 4: Chemical tree with 11 vertices and presented in [14] and [22] as second-minimum for the Randić index.

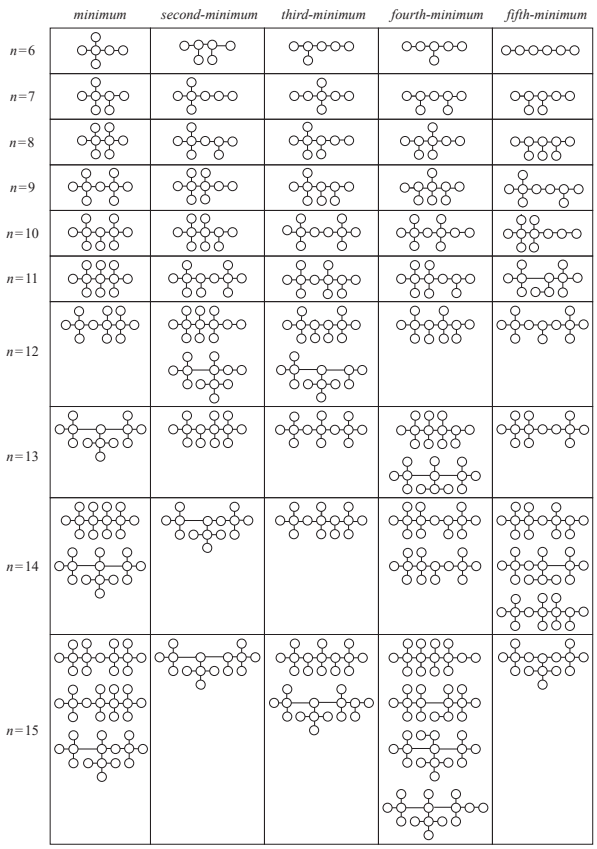

Figure 5: Extremal chemical trees for the Randić index.

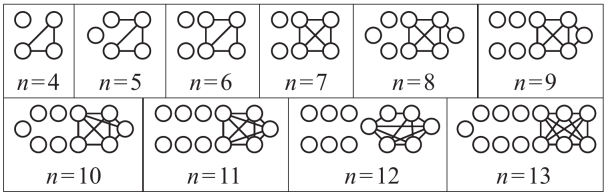

Figure 6: Graphs with minimum Randić index and $n-1$ edges.

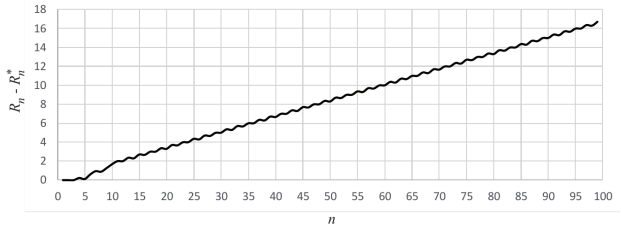

Figure 7: HERE Price of connectivity for the Randić index of chemical trees.

edges, $4 \leq n \leq 13$ vertices, and minimum Randić index.

Let R_n be the minimum Randić index of a chemical tree with n vertices, and let R_n^* be the minimum Randić index of a simple graph with n vertices, $m-1$ edges and maximum degree $r \leq 4$. Clearly, $R_n \geq R_n^*$. The difference $R_n - R_n^*$ is somehow a *price of connectivity* [6] which we represent in Figure 7 for $n \leq 99$. The curve indicates a regular shape for all $n \geq 11$. By analysing the extreme graphs for R_n^*, we have observed that they all have $\frac{n-1}{2}$ vertices of degree 4, and $\frac{n+1}{2}$ isolated vertices if n is odd, and $\frac{n-2}{2}$ vertices of degree 4, 1 vertex of degree 2, and $\frac{n}{2}$ isolated vertices if n is even. The regular shape of the curve in Figure 7 is due to the fact that for all $n \geq 11$, we have

$$R_n - R_n^* = \begin{cases} \frac{n}{6} & \text{if } n \mod 6 = 0 \\ \frac{n-1}{6} + \frac{\sqrt{3}-1}{2} & \text{if } n \mod 6 = 1 \\ \frac{n+4}{6} - \frac{\sqrt{2}}{2} & \text{if } n \mod 6 = 2 \\ \frac{n-3}{6} + \frac{\sqrt{2}}{2} & \text{if } n \mod 6 = 3 \\ \frac{n-4}{6} + \frac{1+\sqrt{3}-\sqrt{2}}{2} & \text{if } n \mod 6 = 4 \\ \frac{n+1}{6} & \text{if } n \mod 6 = 5. \end{cases}$$

As final illustration of the use of the proposed methods, we give in Figure 8 all simple chemical trees with $6 \leq n \leq 12$ vertices having minimum, second-minimum, third-ninimum, fourth-minimum, and fifth-minimum value of the second Zagreb index.

While this was not the case for the Randić index, it happens several times that an extremal value of the second Zagreb index in reached with more than one

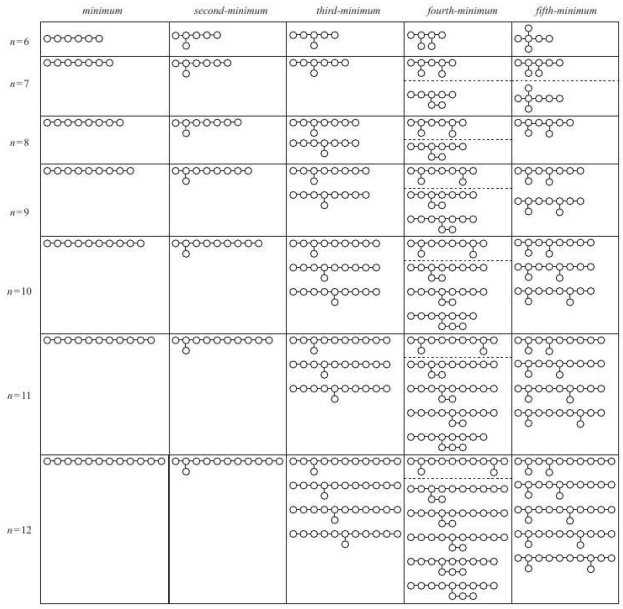

Figure 8: Extremal chemical trees for the second Zagreb index.

M-matrix. Extreme graphs having the same value, but different M-matrices are separated with a dotted line in Figure 8. For example, for $n = 10$, there are 4 graphs with fourth-minimum value of the the second Zagreb index. The first one was obtained from a first M-matrix, while the three others were obtained from a second M-matrix.

Conclusion

We have given necessary and sufficient conditions on the numbers m_{ij} of edges with end-degrees i and j for the existence of a simple graph or a simple connected graph with fixed maximum degree. These conditions can be imposed by an integer programming model, and graphs with these m_{ij} values can be generated using the proposed algorithms.

We have shown that these models and algorithms are very helpful to determine all extremal graphs of Adriatic indices that linearly depend on the the n_i and m_{ij} values.

References:

[1] M. Aouchiche. *Comparaison automatisée d'invariants en théorie des graphes*. PhD Thesis, École Polytechnique de Montréal, February 2006.

[2] M. Aouchiche, J.-M. Bonnefoy, A. Fidahoussen, G. Caporossi, P. Hansen, L. Hiesse, J. Lacheré and A. Monhait. *Variable neighborhood search*

for extremal graphs. 14. The AutoGraphiX 2 system. In L. Liberti and N. Maculan (editors), Global Optimization: From Theory to Implementation, Springer (2006) 281–310.

[3] M. Aouchiche, G. Caporossi and P. Hansen, *Variable neighborhood search for extremal graphs 20. Automated comparison of graph invariants.* Match Commun. Math. Comput. Chem. 58 (2007) 365–384.

[4] Aouchiche M, Caporossi G, Hansen P, Lucas C (2013) Variable Neighborhood Search for Extremal Graphs. 28: AutoGraphiX After Fifteen Years Les Cahiers du GERAD, G-2013-12.

[5] E. K. Burke and G. Kendall, *Search methodologies. Introductory tutorials in optimization and decision support techniques.* Springer, Berlin, 2005.

[6] E. Camby, J. Cardinal, S. Fiorini, O. Schaudt, The price of connectivity for vertex cover, Discrete mathematics and theoretical computer science 16 (1), 207–224 (2014).

[7] G. Caporossi and P. Hansen, *A learning optimization algorithm in Graph Theory. Versatile Search for extremal graphs using a learning algorithm.* Lecture Notes in Computer Science 7219 (2012) 16–30.

[8] G. Caporossi and P. Hansen. *Variable neighborhood search for extremal graphs: V. Three ways to automate finding conjectures.* Disc. Math, 276 (2004) 81–94.

[9] G. Caporossi and P. Hansen, *Variable neighborhood search for extremal graphs: I. The AutoGraphiX system.* Disc. Math. 212 (2000) 29–44.

[10] G. Caporossi and P. Hansen, *Finding relations in polynomial time*. In Proceedings of the XVIth International Joint Conference on Artificial Intelligence (Stockholm, 1999), vol. 2, 580–585.

[11] J. Christophe, S. Dewez, J.-P. Doignon, G. Fasbender, P. Grégoire, D. Huygens, M. Labbé, S. Elloumi, H. Mélot and H. Yaman, *Linear inequalities among graph invariants: using GraPHedron to uncover optimal relationships.* Networks 52 (2008) 287–298.

[12] F. Glover and G. Kochenberger, (Eds.), *Handbook of metaheuristics.* Kluwer, Amsterdam, 2003.

[13] I. Gutman and N. Trinajstić, Graph theory and molecular orbitals. Total ρ-electron energy of alternant hydrocarbons, Chemical Physics Letters 17 (4) 535–538 (1972).

[14] I. Gutman, O. Miljković, G. Caporossi and P. Hansen, Alkanes with small and large Randić connectivity indices, Chemical Physics Letters 306 (5) 366–372 (1999).

[15] P. Hansen, A. Hertz, C. Sellal, D. Vukičević, M. Aouchiche, G. Caporossi, *Edge Realizability of Connected Simple Graphs.* MATCH Commun. Math. Comput. Chem. 78 (2017) 689–712.

[16] P. Hansen and N. Mladenović, *Variable neighborhood search.* In [12], 145–184.

[17] P. Hansen and N. Mladenović, *Variable neighborhood search: principles and applications.* European Journal of Operational Research, 130 (2001) 449–467.

[18] P. Hansen and N. Mladenović, *Developments of variable neighborhood search.* In C. Ribeiro and P. Hansen (Eds.), *Essays, surveys in metaheuristics.* Kluwer, Amsterdam, 2001, 415–440.

[19] P. Hansen and N. Mladenović, *An introduction to variable neighborhood search.* In S. Voss *et al.* (Eds.), *Metaheuristics, advances, trends in local search paradigms for optimization*. Kluwer, Amsterdam, 1999, 433–458.

[20] P. Hansen, N. Mladenović and J. A. Moreno Pérez, *Variable neighborhood search: methods and applications.* Ann. Oper. Res. 175 (2010) 367–407.

[21] S. C. D. Kirkpatrick, Jr. Gellatt and P. Vecchi, *Optimization by simulated annealing.* Science 220 (1983) 671–680.

[22] X. Li and I. Gutman, Mathematical aspects of Randić-type molecular structure descriptors, *Mathematical Chemistry Monographs*, University of Kragujevac, 2006.

[23] N. Mladenović and P. Hansen, *Variable neighborhood search.* Computers and Operations Research, 24 (1997) 1097–1100.

[24] M. Randić, Characterization of molecular branching, Journal of the American Chemical Society 97 6609–6615 (1975).

[25] C. R. Reeves, (Ed.), *Modern heuristic techniques for combinatorial problems.* Blackwell Scientific, Oxford, 1993.

[26] N. Sloane, The on-line encyclopedia of integer sequences. Available at *http://www.research.att.com/minjas/sequences/.*

[27] D. Vukičević and M. Gašperov, Bond additive modeling 1. Adriatic indices, Croatica Chemica Acta 83 243–260 (2010).