Les Cahiers du GERAD ISSN: 0711–2440

How Far Is, Should and Could Be Conjecture-Making in Graph Theory an Automated Process ?

Pierre Hansen

G-2002-44

August 2002
Revised: August 2003
Draft. Do not cite without the author's permission.

Les textes publiés dans la série des rapports de recherche HEC n'engagent que la responsabilité de leurs auteurs. La publication de ces rapports de recherche bénéficie d'une subvention du Fonds F.C.A.R.

How Far Is, Should and Could Be Conjecture-Making in Graph Theory an Automated Process ?

Pierre Hansen

GERAD
and
École des Hautes Études Commerciales
Montréal

August, 2002
Revised: August, 2003

Les Cahiers du GERAD

G–2002–44

Copyright © 2002 GERAD

Draft. Do not cite without the author's permission.

Abstract

Computer-assisted and automated conjecture-making in graph theory is reviewed, focusing on the three operational systems GRAPH, Graffiti and AutoGraphiX (AGX). A series of possible enhancements, mostly through hybridisation of these systems, are proposed as well as several research paths for development of the area.

Keywords: graph, conjecture, computer-assisted, automated.

Résumé

On passe en revue la génération de conjectures assitée par ordinateur et automatisée en théorie des graphes, en considérant plus particulièrement les trois systèmes opérationnels GRAPH, Graffiti et AutoGraphiX (AGX). Une série d'améliorations possibles, le plus souvent par hybridation de ces systèmes, sont proposées ainsi que plusieurs voies de recherche pour le développement du domaine dans son ensemble.

Mots clés: graphe, conjecture, assisté par ordinateur, automatisé.

1 Introduction

1.1 Conjectures

Roget's New thesaurus [147] defines *conjecture* as

> "a judgment, estimate or opinion arrived at by guessing: guess, guesswork, speculation, supposition, surmise".

So, uncertainty is stressed. In mathematics, the word "conjecture" has a more precise meaning. In their *Dictionary of Mathematics*, Bouvier and George [22] define it as follows:

> "Conjecture: An *a priori* hypothesis on the exactness or falseness of a statement of which one ignores the proof".

Knowledge should back this hypothesis, and make the conjecture valuable, as stressed by Mac Lane [128]:

> "Conjecture has long been accepted in mathematics, but the customs are clear. If a mathematician has really studied the subject and made advances therein, then he is entitled to formulate an insight as a conjecture, which usually has the form of a specific proposed theorem. Riemann, Poincaré, Hilbert, Mordell, Bieberbach, and many others have made such deep conjectures".

Further examples of important conjectures, this time in graph theory, are the four-color conjecture [149], proved in 1976 by Appel and Haken [7] [8] [9] (see also the more recent proof of Robertson *et al.* [146]) and the strong perfect graph conjecture of Berge [16] [17] very recently proved by Chudnovsky *et al.* [47] [48]. The former may have initially been a happy guess of a student, Francis Guthrie, but was popularized by a major mathematician, Augustus de Morgan. Its (correct) proof took 125 years and was computer-assisted. No proof without partial automation is known. The latter was proposed in 1961 by one of the most prominent graph theorist of the time. It took 41 years and the work of scores of mathematicians to be finally proved (without computer assistance). So, conjecturing supposes knowledge and insight. Guessing is easy but conjecturing is hard; we will see that this holds for computers as for humans.

1.2 Automation

Again according to Mac Lane [128], the sequence for the understanding of mathematics may be:

> "Intuition, Trial, Error, Speculation, Conjecture, Proof".

Proof is the ultimate goal and has attracted the most attention, including in attempts to automate mathematics. Yet, it is far from the whole story. Hardy [114] reminds us that

> "All physicists and a good many quite respectable mathematicians are contemptuous about proof".

Discovering interesting or beautiful conjectures, even if someone else proves (or refutes) them, is of importance.

Clearly, theorems are first conjectures, possibly known as such only to those who prove them. Often, only the final result, i.e., the theorem, is published. The discovery process is not explained, and further discoveries may be made more difficult than necessary. A few mathematicians and philosophers of science have focused on this process. Prominent among them are Euler, Polya [138] [139] and Lakatos [123]. Recent studies of the application of Popper's ideas in mathematics [140] and their development are also of interest [94] [95].

Automated theorem proving is a well-developed field, with numerous researchers, tens of books and a rapidly increasing record of successes [162]. A good example is the recent 16-line automated proof by Mc Cune [133] of the Robbins conjecture:

"All Robbinsonian algebras are Boolean algebras",

which had been open for 63 years.

In graph theory, only simple propositions can at present be proved in an entirely automated way (see Section 2 for a brief discussion). Computer-assisted proofs, mostly based on enumeration routines, are becoming common. To illustrate, the survey of Radzizowski [145] on small Ramsey numbers mentions computer-assisted results from 71 papers.

In contrast, computer-assisted and automated conjecture-making in mathematics, the mathematical branch of discovery science, has attracted few researchers up to now, despite some notable successes. Outside of graph theory one may mention the important work of several mathematicians on integer relation detection [115] [11] [21]. This led, among other applications to Apéry-like formulae for $\zeta(4n+3)$, new Euler sums and formulae for various constants including one for π with the astonishing consequence that one can compute, in base 2, the digits of π beginning at any place (e.g. from the trillionth' one) without knowing the previous ones. While the important work of Wu [163] [164], and Chou *et al.* [42] [44] [45] in plane geometry is mostly aimed at automating proofs, it includes conjecture-making routines. This led to the discovery of new families of Pascal conics [44]; recently a procedure for finding *all* relations implied by a given configuration of lines and curves in the plane has been obtained [45]. Hájek and Havránek [100] [101] and Hájek and Holeňa [102] have studied mathematical formulations for a general theory of the mechanization of hypothesis formation. They introduce formal logics for that purpose and may have been one of the sources of inspiration for the system GRAPH discussed below.

Graph-theoretic work will be discussed throughout this paper. This will be done by a study and discussion of some of the best developed systems, no general mathematical framework for making conjectures in graph theory being in current use. But a preliminary question should be addressed:

How far should conjecture-making in graph theory be automated?

Langley [121] comments as follows on discovery science systems:

"Although the term *computational discovery* suggests an automated process, close inspection of the literature reveals that the human developer or user plays an important role in any successful project. Early computational research on scientific discovery downplayed this fact and emphasized the automation aspect in general keeping with the goals of artificial intelligence at the time. However, the new climate in AI systems that advise humans rather than replace them, and recent analyses of machine learning applications. . . suggest an important role for the developer".

Two things should be distinguished here: on the one hand, that knowledge due to the developer, and possibly many others (e.g. numerous algorithms for computing graph theoretic invariants) is embedded in the system appears to be necessary to obtain conjectures; on the other hand, that the user may interact or not with the system, leading to computer-assisted or to automated discoveries.

In graph theory, three additional reasons may be adduced for preferring computer-assisted systems to automated ones:

(i) the difficulty of automation may limit the scope of problems addressed;

(ii) the ultimate goal being proof, interaction with the system is more likely to lead to insights about how to prove the conjectures found than just reading their statement;

(iii) such interaction may also be very fruitful from the pedagogical point of view. This question will be discussed further in Section 3.

However, automating a system for making conjectures in graph theory is a challenge, and may lead to original ways of addressing this problem. Moreover, comparison of this work with the treatment of similar problems within close fields such as automated theorem proving or data mining, may foster cross-fertilization.

This author's view is that both computer-assisted and automated conjecture-making are of interest; in this paper, the main focus is on the latter.

1.3 Definitions

We will adopt the following terminology: an *automated system* will be synonymous with a *fully* automated one, and this means that

(i) *input should be limited to the problem statement* which implies further information on the problem or closely related ones cannot be introduced *at that time*, but may of course already belong to one or another of the systems databases;

(ii) *there should be no human intervention between problem statement and output of the results*;

(iii) *output of the results should be the final step*, which implies there should be no human selection of those conjectures about the problem under study which are publicized; of course the users are then free to choose those they will try to prove.

Otherwise, the system will be called *computer-assisted*.

This is in keeping with usual practice. To illustrate points (i) and (ii), "Deep Blue" [30] [118] is an automated system which includes considerable knowledge about chess-playing (and Kasparov's way to play chess) due to the developers. In competition it can be tuned before a game but not when this game is in process, and it only receives notice of the opponent's moves [118].

To illustrate point (iii) observe that some researchers (e.g. [120]) claim that computers can compose poetry and try to make their point by selecting among a large output, obtained by their system from some poets vocabulary, usually very short "poems" which appear to make sense. If one generates a sufficiently large number of "poems" and selects drastically among them, this is bound to work (to some extent), but the conclusion is far from clear.

When an automated system makes conjectures we will say they are obtained *by* the system; when a computer-assisted system does so, we will say the conjectures are obtained *with* the system.

Refuting or corroborating conjectures known beforehand with a computerized system will be referred to as *testing* them; conjectures which are corroborated may be improved (e.g. stronger bounds may be considered) and this will be called *strengthening* them; finding new conjectures will be called *conjecture-making*, and can be unassisted (or done by hand), computer-assisted or automated.

To the best of our knowledge, present systems for conjecture-making in graph-theory are either computer-assisted or can be used both in computer-assisted mode and, in rare cases, in automated mode. The question of whether one computer-assisted system is more automated than another cannot be answered in a clear-cut way as different systems perform different tasks. Therefore, we will describe these tasks, state which of them are automated and how, which are not and how they are done, and let the reader judge.

About half a dozen systems for conjecture-making in graph theory and other close purposes have been developed. We distinguish between *experimental* and *operational* systems. An experimental system explores an idea, without necessarily leading to new results (or to just a few, due to its developers); its aim is often to understand the way mathematicians reason or to help them in various tasks. Such systems, while they may be inactive for the time being, have potential, particularly in conjunction with others, as discussed briefly in various places of this paper. They include:

(a) the *INGRID* system of Brigham and Dutton [25] [26] [27] [28], which manipulates formulae on graph invariants from a database to compute bounds on some invariants when others are limited to some range. INGRID can be used to

   (i) help solve practical problems,

   (ii) derive new theorems (by selecting relations leading to them),

   (iii) test the effectiveness of new theorems (by showing they are or not consequences of one or several previously known ones),

   (iv) test conjectures (viewed as "temporary theorem" to see if this implies some contradiction),

   (v) resolve open problems (by showing they imply some contradiction), and

   (vi) help to study graph theory.

   As explained in [113], some of these functions may be viewed as obtaining particular types of conjectures.

(b) the *graph theorist* system of Epstein [73] [74] [75] [76]; this knowledge intensive, specific domain learning system uses algorithmic descriptions of classes of graphs such as connected, acyclic, bipartite and so forth. It mainly uses theory-driven discovery of concepts, conjectures and theorems, based upon search heuristics, but also infers explanations from factual input about graphs.

There are three main operational systems:

(a) the GRAPH system, developed by Cvetković and co-workers [59] [60] [54] [61] [55] [56] [62] [63], which pioneered the man-machine type of research in graph theory. Built between 1980 and 1984 this system was extensively used to find conjectures and prove theorems in graph theory (usually the latter only being published), with an emphasis on algebraic graph theory. Cvetković and Simić [64] review 92 papers by 23 authors on GRAPH, its uses and results obtained with it from 1982 onwards. GRAPH comprises

  (i) a bibliographic component, BIBLI,

  (ii) an algorithmic component, ALGOR, and

  (iii) an automated theorem proving one, THEOR.

(b) the Graffiti system, due to Fajtlowicz [81] [80] [82][83] [84] [85] [79] [87] and developed since the mid-eighties, with from 1990 onwards collaboration of De La Vina, notably in the development of its DALMATIAN version. This system generates a large number of *a priori* conjectures, under the form of algebraic relations between graph invariants, then selects among them, by eliminating false or uninteresting conjectures through testing them on a database of graphs, applying heuristics and building counter-examples. Conjectures which pass these correctness and interestingness tests are proposed, after further selection, to the mathematical community in the large email file “Written on the Wall” which is updated from time to time. More than 70 mathematicians, among them some famous ones, sent proofs, or refutations of those conjectures, listed in that file. Many papers on proofs, and more often disproofs, sometimes with corrected results which led to further developments or strengthened conjectures, have been published. De La Vina [67] lists 75 such papers, technical reports and theses from 1986 onwards.

(c) the AutoGraphiX (AGX) system, due to Caporossi and Hansen [37] [31] [65] [34] [35] [106] [33] [6] [32] [36] [107] which generates many extremal or near-extremal graphs for some invariant or formula involving several invariants, then derives various results from them. This system may be used to

  (i) find a graph satisfying given constraints;

  (ii) find optimal or near-optimal values for a graph invariant on a family of graphs with given constraints;

  (iii) refute, corroborate or strengthen a conjecture;

  (iv) make a conjecture in computer-assisted or automated mode;

(v) suggest ideas of proof.

A series of papers on the system, its uses, results and comparative performance have been published. Aouchiche [5] lists 40 papers on AGX and its results, or related to its results, published since 1999, submitted, or to appear.

Collectively, this number of papers (over 200) is among the largest in the field of discovery science.

Some programs from graph theory not designed specifically for making conjectures may be useful to do so, either on their own or in conjunction with others. This is the case in enumeration where e.g. programs such as *Nauty* and *geng* of McKay [131] [132] helped to conjecture and then determine many Ramsey numbers.

Conjectures can also be obtained by serendipity. As explained in more detail in [104], a program for coloring planar graphs written in Mathematica by Wagon, always used 3 colors when applied to rhombic Penrose tilings; Sibley and Wagon [152] then proved 3 colors suffice, a problem that had been open for 20 years. Another example relies on a program from mathematical programming: a mixed-integer formulation of the problem of determining the Clar number of a benzenoid [110], due to Hansen and Zheng [111], never used branching. The conjecture that linear programming sufficed to solve this problem was later proved by Abeledo and Atkinson [1] [2].

1.4 Plan of the paper

This paper has two complementary aims:

(i) *assess the state-of-the art* in computer-assisted and automated conjecture-making in graph theory. This will be done in the next three sections, devoted respectively to GRAPH, Graffiti and AGX, with special emphasis on their conjecture-making functions;

(ii) *make a series of proposals for advancement* of this field. They will be interspersed in the next three sections and will take two forms. First, *Proposed Enhancements* (PE) will suggest ways to improve specific steps or functions of the system under study; they will often be suggested by ways to solve similar problems in other systems and the suggestions will then amount to hybridizing them. Second, *Research paths* (RP) will draw attention upon open problems or general questions related to conjecture-making in graph theory, as well as links to establish with other domains of research. They are often long-term goals, sometimes quite speculative. Separation between study of systems and proposals will be indicated by numbering them PEk or RPk, with a $\Box$ sign as the end of the corresponding statement.

The three operational systems GRAPH, Graffiti and AGX will be studied in sections 2, 3 and 4 respectively. Conclusions will be drawn in Section 5.

2 Graph

As mentioned in the introduction, GRAPH has three components, BIBLI, ALGOR and THEOR. ALGOR is the most directly related to conjecture-making but both BIBLI and THEOR bear upon problems of importance for conjecture-making systems too. So we examine all three of them in turn.

2.1 BIBLI

The GRAPH system uses a formalized subset of the everyday English language, called Graph Theoretic Computer Language. It is described in [59]. It is an interactive language used from a terminal keyboard; in recent versions a mouse can be used also for some operations.

The BIBLI component is devoted to bibliographic data processing: it allows storage and retrieval of information on papers, books, proceedings, reports, abstracts, manuscripts and documents. Its functions, rarely available at the time of inception, are now in wide use in systems accessible on the web such as *Google, Web of Science* or *Citeseer*, but it remains useful for tailor-made bibliographies such as that one of the book of Cvetković *et al.* "*Recent Results in the Theory of Graph Spectra*" [57].

While very large amounts of data are now available online and special sites devoted to graph theory, such as the *Graph Theory White Pages* are open to the general public, the documentation problem in graph theory is far from solved (All those who have painstakingly derived a series of conjectures, transformed them by proof into theorems only to find in a last check most or all results to be known but expressed in a different language are well aware of this problem). Indeed, the graph theory literature is vast, dispersed over many fields, growing in a savage way and, as a consequence, terminology is far from unified. Moreover, due to dependence between concepts, the same results can take different forms e.g. in the graph G or its complement $\bar{G}$, or after eliminating one or another invariant by a linear equation such as those of Gallai's theorem [92]. Finally, some results can be expressed in different ways because concepts have a nonlinear dependence. To illustrate, even if one knows that the Wiener index of a tree T [72] is another name for the sum of distances between pairs of vertices of T, one might miss equivalent results expressed in terms of average distance between pairs of distinct vertices of T.

Brigham and Dutton [26] [27] have gathered 458 relations between graph invariants, used in their system INGRID. They can help in checking whether a result is new, but if this has to be done with a chance of success, a much more comprehensive system should be built, in a collaborative effort, similar to that which gave rise to Sloane's On-line Encyclopedia of Integer Sequences [157]. The following research paths sketch how this might be done:

RP1. *Find linear equality relations between graph invariants.* Consider a large number of graph invariants and programs to compute them (available in the cited systems, in Graphbase [119] or LEDA [134] and on the Web). Compute values of these invariants for a large set of graphs. Then use the numerical relation-finding routine of AGX (see Section 4) to obtain a basis of affine relations on these invariants. If some new relations are found, prove them. □

RP2. *Define a standard set of invariants* in terms of which all others will be expressed and (one or several) *standard forms for relations in graph theory.* Write a translator program which will express (as far as possible) any formula in standard form and conversely express a standard-form formula in one or all equivalent forms. Programs for algebraic manipulations such as Mathematica [161] or Matlab [130] might be used for that purpose. □

RP3. *Organize a site* for interactive addition to and consultation of a database of graph theory relations. These relations might be valid for all graphs, or for important families of subgraphs, e.g. bipartite, triangle-free, of girth at least 5, and so forth. □

Another important open problem, related to storing graph theory relations is to find if a given relation is redundant, i.e., implied by one or more relations already in the database. This can be done by finding a graph within a database for which it is not the case, as in the DALMATIAN version of Graffiti [84] (see Section 3) or in an algebraic way as in INGRID [28] or by showing that the relation is not best possible (assuming a best possible relation is known).

Given invariants $i_1, i_2, \ldots, i_p$ of a graph G one can define, as in [109], a *canonical form* for relations involving these invariants as

$$i_k \leq f(i_1, i_2, \ldots, i_{k-1}, i_{k+1}, \ldots, i_p) \tag{2.1}$$

or

$$i_k \geq g(i_1, i_2, \ldots, i_{k-1}, i_{k+1}, \ldots, i_p); \tag{2.2}$$

such relations are *sharp* (or best possible) if for all values of $i_1, i_2, \ldots, i_{k-1}, i_{k+1}, \ldots, i_p$ compatible with the existence of a graph there is a graph such that the relation is satisfied as an equality. A set of canonical relations is *complete* if the $2p$ relations (2.1) and (2.2) on $x_1, x_2, \ldots, x_p$ are sharp. One such set for the three parameters $\alpha(G)$ (independence number), n (order) and m (size) is given in [109]. The relation [105] [88]

$$\alpha(G) \geq \left\lceil \frac{2n - \frac{2m}{\lceil \frac{2m}{n} \rceil}}{\lceil \frac{2m}{n} \rceil + 1} \right\rceil \tag{2.3}$$

is sharp, while the following one, derived from Turan's theorem [159],

$$\alpha(G) \geq \frac{n^2}{2m + n} \tag{2.4}$$

is not. It is thus redundant but might be kept also if one is more interested in simplicity than in sharpness. Observe also that if a sharp relation is known one might consider that it is not useful to compare it to another one, yet the latter could also be sharp and simpler as is the case for (2.3) which is equivalent to but simpler than the relation given in [105].

2.2 ALGOR

This part of the system GRAPH is directly connected to conjecture-making ([59], p20):

> "The part of the system "GRAPH" described is primarily meant as a means for quick[ly] checking, disproving or making conjectures in graph theory. Facilities provided by the system enable to get the answer on a great number of questions on graphs of a reasonable size in a few seconds (of course, what does a reasonable size mean depends on the problem considered)."

Also:

> "Another situation in which the system can help is the following. Many results in graph theory begin with an observation which proves the desired statement for all but a finite number of graphs. These exceptional graphs are, as a rule, of a small size. The next part of the proof consists then in checking whether the statements hold for these graphs and that can be performed with the help of the system."

ALGOR solves a series of problems on particular graphs. They can be divided as follows: ([59], p11):

(a) manipulative tasks (setting and displaying values of the mentioned objects (i.e., graphs, values of the type integer, real and complex, and families of sets of integer values),
(b) creating common graphs (e.g. complete graphs, circuits, etc) or random graphs,
(c) creating graphs by performing graph-theoretic operations (e.g. complement of a graph, product of two graphs, etc),
(d) relabelling (points or lines of) graphs (by given permutations, at random, etc),
(e) determining integer invariants in graphs (e.g. number of some subgraphs, order of some point, etc.),
(f) determining real invariants of a graph (e.g. eigenvalues, eigenvectors, etc),
(g) checking properties of graphs, (e.g. whether a graph is planar or hamiltonian, whether two graphs are isomorphic, etc),
(h) listing families of graph characteristics (e.g. point degrees, components; etc).

Each group of operations is characterized by a verb in the commands used. They have a simple and transparent form, e.g.

CREATE $<$ g-name $>$ [AS] $<$ type of graph $>$ [OF] [ORDER] $<$ integer $>$,

for instance:

CREATE $G1$ CIRCUIT OF ORDER 12

or

FORM [g-name] [AS] [THE] $<$ integer $>$ [TH] $<$ operations $>$ [GRAPH]

[OF] $<$ g-name $>$,
for instance:

FORM H AS THE 4TH SUBDIVISION GRAPH OF G

The operations are: DISTANCE, (PATH), POWER, SUBDIVISION, (TRAIL), (WALK). Names in parentheses correspond to operations not yet implemented when [53] was written.

PE1. Complete GRAPH by enriching its functions as planned. This task is in progress in the system NEWGRAPH, currently developed. □

Determining invariants is broken down in four categories:

(a) Invariants of the graph
(b) Invariants of point of the graph
(c) Invariants of a given size
(d) Invariants of two points of the graph

Commands for invariants of a graph have two forms:

(i) determining the number of objects in the graph, which can be (AUTOMORPHISMS), BLOCKS, BRIDGES, CENTRAL POINTS, (CIRCUITS), CLIQUES, (COCLIQUES), COMPONENTS, CUTPOINTS, (INDEPENDENT LINES), LINES, LOOPS, MAXDEGREE, MINDEGREE, (ORBITS), PENDANT LINES, (PENTAGONS), POINTS, QUADRANGLES, TRIANGLES;
(ii) computing the value of an invariant such as (CHROMATIC CLASS), (CHROMATIC INDEX), CHROMATIC NUMBER, CIRCONFERENCE, (CLIQUE NUMBER), (COARSENESS), (COMPLEXITY), (CROSSING NUMBER), CYCLOMATIC NUMBER, (DETERMINANT), DIAMETER, (EXTERIOR STABILITY), (GENUS), GIRTH, (INTERIOR STABILITY), (LINE CONNECTIVITY), (PERMANENT), (POINT CONNECTIVITY), RADIUS, RANK, (THICKNESS).

For instance:

DETERMINE THE NUMBER OF TRIANGLES OF G,
DETERMINE DH THE DIAMETER OF H.

A point invariant such as DEGREE, ECCENTRICITY, etc would be found by making a command such as

DETERMINE DEGREE OF 7 OF G

where 7 is the label of a point. Commands for invariants involving two points or real invariants of a graph are similar. Possible objects are (CIRCUITS CONTAINING), COMMON NEIGHBOURS, (DISJOINT PATHS), DISTANCE, LINE LABEL, LINES INCIDENT, (PATHS), (TRAILS), (WALKS) in the former case and (ADMITTANCE SPECTRUM), (ANGLES), BOND ORDERS, CHARGES, DISTANCE, INDEX, (DISTANCE SPECTRUM), EIGENVALUES, EIGENVECTORS, ENERGY, (MAIN ANGLES), (R-SPECTRUM), SEIDEL SPECTRUM in the latter.

The GRAPH system can check many properties of graphs such as ACYCLIC, BIPARTITE, BLOCK, (BLOCK CUTPOINT GRAPH), (BLOCK GRAPH), CIRCUIT, (CLIQUE GRAPH), complete, connected, (cutpoint graph), eulerian, forest, hamiltonian, hypohamiltonian, (interval graph), line graph, loopless, (moore graph), (outerplaner), (perfect), planar, (prime), (selfcomplementary), (selfdual), semi regular, (semitotal line graph), (semitotal point graph), strongly regular, (subdivision graph), (total graph), totaly disconnected, (traversible), tree, triangle free, trivial, unicyclic, wheel, without multiple lines.

Commands are for instance

CHECK WHETHER $G1$ IS PLANAR,
CHECK WHETHER $G2$ IS A TREE.

or, for properties of a point of a graph:

CHECK WHETHER THE POINT 5 IS ISOLATED IN G,

or of two graphs

CHECK WHETHER $G1$ AND $G2$ ARE ISOMORPHIC.

Clearly the system GRAPH can answer a large number of questions regarding particular graphs. It can also check for graphs with some property among several lists of graphs, e.g. connected graphs up to 6 points, regular graphs up to 7 points, trees up to 10 points, cubic graphs op to 12 points, etc.

Results of GRAPH consist, as mentioned above, of computer-assisted conjectures, refutations and proofs. Most of the published results are theorems, and while mention of system GRAPH is made, details on how it led interactively to conjectures, refutations or proofs are unfortunately not given except in [59] (automated theorem-proving is discussed in more detail [62] [56]).

We list a couple of results obtained with GRAPH, see [64] for a more comprehensive set. Let G be a graph, v a distinguished vertex, and $N_1(v)$, $N_2(v)$ a partition of the neighbours of v. If G' is obtained from $G - v$ by adding vertices v_1, v_2 and edges $\{v_1, w\}$ with $w \in N_1(v)$ and $\{v_1, w\}$ with $w \in N_2(v)$, G' is obtained by *splitting* vertex v.

The following result was conjectured with the system GRAPH and proved in [153]: *If G is a connected graph and G' is obtained from G by splitting a vertex then* $\lambda_1(G') < \lambda_1(G)$ (where $\lambda_1(G)$ is the *index* of G or largest eigenvalue of its adjacency matrix).

Denote by $\rho(k)$ the largest eigenvalue of the graph obtained from the cycle C_n with $n \geq 6$ by adding an edge between two vertices at distance $k = 2, 3, \ldots, \lfloor n/2 \rfloor$. On the basis of experiments conducted with GRAPH it was conjectured that $\rho(k)$ is monotonous and decreases. This was proved in [148] [154].

2.3 THEOR

The THEOR component of GRAPH is designed for computer-assisted or automated theorem-proving in graph theory, and is described in Cvetković and Pevac [62]. We only discuss it briefly as this paper's topic is not automated theorem proving. Graph theory is formalized using a special first-order predicate calculus, called "arithmetic graph theory" (AGT). It contains point variables, line variables, integer variables, graph names, constraints, function names, operations over graphs and predicates.

The effectiveness of the prover depends largely on a set of lemmas which represent beginner's knowledge of graph theory. The user may select more advanced lemmas.

A resolution-based prover is a subsystem of a natural deduction interactive theorem prover. The interactive prover provides a proof for a given goal sentence P by splitting it into subgoals, which are further split, thus generating a proof tree memorized by the system. This tree is a rooted one, and the user can move the current root, i.e., select the subgoal next considered. He can also inform the system about the truth of a subgoal. The resolution-based prover can be applied to any subgoal and the proof is completed when all subgoals are proved. Subgoals may be processed by case analysis, forward chaining, *reductio ad absurdum*, simplification or extension of the formula, expressing it in an equivalent form, etc.

A completely automated proof of the simple sentence

> "If the graph is connected, then the graph is trivial or there is no point x such that x is isolated"

is obtained and has 10 lines. The sentence

> "If the graph is not connected, then the complement is connected"

is proved interactively, in 38 lines. Further examples are given in [56].

These examples show the difficulty inherent in full formalization of graph theory. Its language, close to English, is deceptively simple. The situation is much easier in logic [162], or in plane geometry where a method of reduction of problems to systems of linear and quadratic equations applies, see e.g. Chou [43]. But as the speed of equally priced computers has augmented since the time GRAPH was developed by a factor of 10^4 to 10^5 and automated theorem proving made much progress, another attempt might be worthwhile.

PE2. Test the automated theorem proving approach of THEOR with a modern computer and a prover such as OTTER [136]. □

Should this attempt be successful, it should meet a wish of Fajtlowicz [84]:

> "the problem of trivial conjectures could be solved if we had automated theorem provers capable of proving the easiest conjectures of Graffiti . . . "

3 Graffiti

3.1 Structure

The Graffiti program is discussed in the series of papers "*On conjectures of Graffiti*" [81] [80] [82] [83] [84] a paper "*On conjectures and methods of Graffiti*" [87] as well as in the more recent paper "*Towards fully automated fragments of graph theory*" [85], and a couple of papers of Larson [124] [125]. De La Vina [68] presents the system Graffiti.pc and, very recently, some recollections about early development and use of Graffiti [69]. Conjectures obtained with Graffiti and their status i.e., proved, refuted or open, are listed in [79].

There are many versions of Graffiti, not all of which appear to have been fully documented [69]. The two main ones appear to be the initial version (with a few developments) described in [80] [81] [82] [83] [87] and the DALMATIAN version described in [84] [85] [124] [125] and [68]. In this subsection we list the steps of both of them. These steps will be discussed in detail in the following subsections.

Unfortunately, no complete and precise description of all steps of the process of obtaining conjectures with Graffiti has been provided. Instead, partial and informal descriptions of the automated steps are scattered over a good half-dozen publications; information about the other steps is given similarly, but in much less detail. This makes rational discussion of the Graffiti system and its applications extremely difficult as it must be preceded by a long reconstruction process, i.e., finding what really happened, or happens, from scant and sometimes contradictory information (as e.g. when computing invariants is attributed to Graffiti in one place and to Algernon in others). The paper of De La Vina [68], written after the first version of the present paper was completed, and remarks of an anonymous referee have been very helpful in this reconstruction process.

Graffiti uses two databases; a database of graphs and a database of conjectures. The former contains graphs proposed by the authors or other researchers, which have refuted some conjectures, together with precomputed values for all invariants considered in the system. The latter contains conjectures generated by the system and not refuted or viewed as non-interesting, or possibly in the DALMATIAN version, viewed as non-informative.

Steps of the process of finding conjectures with the initial version of Graffiti appear to be the following:

Step 1. Problem statement: Find relations between a set of invariants $i_1(G), i_2(G)\ldots$ chosen by the user.

Step 2. Conjecture generation: The program generates a set of inequalities of the forms $i_1(G) \leq i_2(G)$, $i_1(G) \leq i_2(G) + i_3(G)$, or similar ones using the selected invariants and possibly small integers (mostly 1).

Step 3. Correctness Test: The program evaluates the inequalities obtained. If one graph refutes them, they are deleted.

Step 4. Heuristic Tests (see below): The program deletes the conjectures which do not pass the test.

Step 5. Counter-example: Find by hand (a) counter-example(s) to at least one of the new conjectures. If one is found, delete the corresponding conjecture.

Step 6. Update of Graph Database: If at least one counter-example has been found compute values of all invariants for the corresponding graph(s). Adds these graphs to the database of graphs and return to Step 3.

Step 7. Elimination of true conjectures: Prove by hand easy new and true conjectures and eliminate them from the database of conjectures (if they are not judged to be interesting).

Step 8. Selection of conjectures: Select, by hand, among the remaining conjectures those considered to be worthy of publication. Make them known, e.g. by including them in the "Written on the wall" file.

Fajtlowicz ([80] p.189) comments as follows on this process, and its interactive character:

"Graffiti makes conjectures by first verifying that it does not know a counter-example to a formula and then by deciding whether the formula makes an interesting conjecture. The first function of the program is highly interactive because a user is expected to find counterexamples to false conjectures and then describe them to the program."

Steps of the process of finding conjectures with the DALMATIAN version of Graffiti appear to be the following:

Step 1. Problem statement: Find lower (or upper) bounds for a user-selected invariant.

Step 2. Conjecture Generation: The program generates an inequality and evaluates the values of both sides of all graphs in the database.

Step 3. DALMATIAN test for informativeness (see below): The program deletes the conjecture if it does not pass the test.

Step 4. Correctness test: The program deletes the conjecture if the inequality does not hold for at least one graph in the database.

Step 5. Other heuristic tests (see below): The program deletes the conjecture if it does not pass one of these tests.

Step 6. Database updating: The program shelves conjectures viewed as less informative due to the addition of the new conjecture.

Step 7. Test for ending conjecture generation: If for each graph in the database of graphs, there is a conjecture for the selected invariant and direction of inequality in the database of conjectures which is sharp (i.e., satisfied as an equality), proceed to the next step. Otherwise, return to Step 2.

Step 8. Counter-example: Find, by hand, a counter-example to one at least of the inequalities generated.

Step 9. Updating database of graphs: If a counter-example has been found, compute with an auxiliary program (Called Algernon) the values of all invariants for this graph, introduce it, together with those values in the database of graphs and return to Step 2.

Step 10. Elimination of true conjectures: Prove by hand easy new and true conjectures and eliminate them from the database of conjectures.

Step 11. Selection of conjectures: Select by hand among the remaining conjectures, those considered to be worthy of publication. Make them known, e.g., by including them in the "Written on the Wall" file.

Note that the correctness test now follows the first interestingness test; the reason appears to be that the DALMATIAN test is quicker than the other one on average.

Observe that as the new conjectures have the same left-hand side invariant and direction of inequality they may be viewed as a system. Note that the procedure described does not necessarily converge (a simple example is given below). It may thus have to be stopped manually, after some time.

At this point, a divergence of opinion between the authors of the Graffiti system and the present author should be clearly stated. Fajtlowicz focuses on what is automated and wishes to limit Graffiti to Steps 1 to 7 above. When they are finished, which constitutes a *round*, the user takes over, does whatever he wishes (eventually with the help of Algernon) and may proceed or not to a further round. So the non-automated part of the conjecture generation process is viewed to be in some sense, outside of Graffiti, while the final conjectures are still attributed to Graffiti alone, as shown by referring to them as "conjectures of Graffiti" or "conjectures obtained by Graffiti".

This author could only accept this view if what is not automated did not substantially affect the final result, i.e., the list of conjectures to be publicized. That steps 8 to 11 play an important role will be documented in the following subsections. Note also that isolating automated parts from the other ones, and giving them a name, then considering the remaining parts to be outside of the process, can lead to a claim that the resulting process is (fully) automated, for any interactive process. The present author cannot agree with such an argument and therefore views Graffiti as a computer-assisted system and not a (fully) automated one. The reader is left to judge.

3.2 Problem statement and generation of *a priori* conjectures.

In the initial version, the problem statement consists in specifying the invariants to be studied (e.g., a set of 20 from the rich library of Graffiti) as well as, possibly, operators such as sum, maximum, minimum, complement etc acting on them, and the desired form of the relations derived. The program then generates systematically such relations.

Forms of conjectures are simple ones, such as $i_1 \leq i_2$ or $i_1 \leq i_2 + i_3$ or sometimes $i_1 + i_2 \leq i_3 + i_4$. Later, ratios were introduced and finally a real algebra on the invariants.

In the DALMATIAN version, the problem statement step has the following form: Find lower (or upper, instead) bounds for a (user selected) invariant. The system then generates a term, as right-hand side of the inequality. This term is obtained by selecting invariants and performing unary or binary operations on them. Examples of such operations are the reciprocal, the natural logarithm, ceiling, addition and multiplication [68].

Details on how this is done, i.e., how many invariants and operations are chosen, within which set, according to which rules and whether or not there is any further user intervention before the session or at the moment the user states his query, are not given. As a consequence, results of Graffiti cannot be reproduced by other researchers.

As the conjecture-generation step conditions the results obtained, it should be analyzed carefully.

First, one may note that the system does not *at this stage*, use any knowledge of graph theory at all, so one should speak of *guesses* rather than *conjectures* (that the subsequent process, which uses graph theoretic algorithms as well as heuristics transforms or not these guesses into conjectures by its selection process will be the crucial point).

In view of this lack of knowledge, one may expect that initially

(i) many conjectures will be false;

(ii) many conjectures will be true but trivial;

(iii) if a very large number of conjectures are generated some of them may be interesting.

Reading all papers written on Graffiti and its conjectures suggests that all three propositions, including the redeeming third one, are true. Fajtlowicz comments as follows on trivial conjectures ([81], p.113) obtained with the initial version of Graffiti. "The number of conjectures, particularly those which are completely trivial, is the main problem and more than half of the program consists of various heuristics whose purpose is detection of trivial and otherwise non-interesting but true conjectures". As documented below in the subsection on selection of conjectures, a substantial number of the selected ones remains false with the initial version and also, to a lesser extent, with the DALMATIAN one.

Second, generation of some important formulae may be, in practice, out of reach of Graffiti, even if the necessary invariants and operations are available, because their algebraic expression is too complex. To illustrate, consider again the bound (2.3) on the stability number $\alpha(G)$. It implies only 2 invariants, m and n, but 12 product, division, sum, subtraction or upper bound operation. The probability that the right invariants and operations, as well as their order can be found *a priori* must be extremely small.

Consequently, Graffiti is not a good tool for obtaining strongest conjectures, i.e., graph theoretical bounds which are best possible in the strong sense, that is, as formula (2.3), tight for all m and n. That other systems, together with a few algebraic manipulations, can do so is illustrated in [107] for the case of an upper bound on the irregularity of a graph.

A related problem arises if the formulae have numerical coefficients; Graffiti introduces a few, usually small, integers. However, if the coefficients are real ones, the number of possible formulae is infinite even in the linear case. How could Graffiti guess *a priori* the right values in such a case?

Third, observe that no computer is needed to generate systematically relations between graph invariants: the (tedious) task of writing down $i_1 \leq i_2, i_1 \geq i_2, i_1 \leq i_3$ and so on can be done by hand without any difficulty; enumerating relations with more complicated forms as done in the DALMATIAN version is only slightly more complicated. Programming this task is also easy.

Fourth, while some *a priori* conjectures are simple and appealing, more complicated ones might not be attractive. To illustrate, the formulae

$$\bar{l}(G) \leq \alpha(G) \qquad \text{(Graffiti 2)}$$

where $\bar{l}(G)$ denotes the average distance between distinct vertices of G and

$$r(G) \leq \alpha(G) \qquad \text{(Graffiti 0)}$$

where $r(G)$ denotes the radius of G, or minimum over all vertices of the largest distance to another vertex, have attracted mathematicians and led to several papers; contrarywise, most mathematicians might consider that the conjecture

"The minimum of derivative of eigenvalues of the gravity matrix is $\leq n/$average distance" (Graffiti 150) is too complicated and specialized.

Fifth, *a priori* conjectures of Graffiti may not have the simplest form they may take. To illustrate the *temperature* t_j of a vertex j is defined by Fajtlowicz as

$$t_j = \frac{d_j}{n - d_j}$$

where d_j is the *degree* of j. The conjecture

$$\bar{l}(G) \leq 1 + \max_j t_j(\bar{G}) \qquad \text{(Graffiti 834)}$$

where $\bar{G}$ is the complementary graph of G, can be reformulated into

$$(1 + \delta(G))\bar{l}(G) \leq n$$

which is simpler, more intuitive, and was refuted [37].

PE3. Add to Graffiti a translation routine which would automatically simplify conjectures. □

3.3 Dalmatian and other heuristics

We now describe and discuss the various heuristics designed to select *interesting* conjectures among those listed *a priori*. The DALMATIAN one [84] is the most recent and apparently also the most powerful. It is based on the notion of *information content* (or informativeness). Basically, a conjecture on an invariant is considered as interesting if and only if it provides some new information for at least one graph in the system's database, i.e. it provides for that graph a strictly better bound than all previous relations. Otherwise, the conjecture is deleted. If it is added to the database of conjectures it may happen that some other conjectures are no more informative and are *shelved*, i.e., kept separately of the database of conjectures (or tagged); if later on some conjecture(s) giving a better or equal bound on i_1 is (are) refuted they can be *unshelved*, or considered as interesting conjecture once again.

Several comments are in order. First, the definition of interestingness on which the dalmatian heuristic is based is *local*, as it depends on the database of conjectures and the database of graphs of Graffiti, and *unstable*, as these databases evolve over time. This implies this definition is not *universal*, i.e., contrary to other mathematical definitions, it cannot be used by all researchers in all places with consistent answers as to whether a conjecture is or not interesting.

Second, the definition may be too *lax*, if the database of conjectures is small or the database of graphs is large (but this would be only temporary as new conjectures are introduced and initial ones shelved), or too *severe* if the database of conjectures is large. Indeed, the situation in which the values of a large set of invariants and many relations on the invariant i_1 under study are known is atypical in graph theory research. Much more often, graph theorists study one invariant as a function of two or three others, ignoring temporarily the other ones.

Four other heuristics were used in early versions of Graffiti. The IRIN heuristic "deletes conjectures which follow from others by transitivity" [81]. The CNCL heuristic deletes conjectures "…in which one invariant on the left is always smaller than an invariant on the right" [81]. The ECHO heuristic [80] applies to conjectures defined for restricted classes of graphs: "its main idea is that a conjecture about a class of objects A is considered noninteresting if it can be generalized to a larger class B ..., the background of A". This heuristic appears still to be used in recent versions of Graffiti. The BEAGLE heuristic is based upon the idea that conjectures involving concepts of a different type are more likely to be interesting [82].

The idea of difference in concept types is related to a representation of concepts as a rooted tree: a graph G is associated with the root and various numerical invariants to its vertices. A concept is a *descendant* of another one if it is computed in terms of that one. The distance between vertices in the tree can be viewed as a distance between the corresponding concepts.

The BEAGLE heuristic removes conjectures involving concepts that are too close; it appears that the DALMATIAN also removes most but not all of them. Larson ([124] p.12) comments on this as follows:

"The BEAGLE heuristic of Graffiti was central to early versions of the program [82]. The function was largely superseded with the introduction of the DALMATIAN heuristic."

Note that the BEAGLE heuristic, as the DALMATIAN one, is defined in terms of the Graffiti system. One may wonder if distance between concepts in graph theory could be defined in a more general mathematical way. This seems to be the case, as lattices of graph theoretic concepts are considered by the Graph Theorist system of Epstein [74] [75] [76] as well as by the Hardy-Ramanujan system of Colton [50] (which is more often applied, however to algebra or number theory than to graph problems). A concept of distance follows. It seems worthy of further study to see to what extent this framework, or more general ones, apply:

RP4. Apply the theory as *formal concept analysis* [93] to graph theory definitions and see if a concept of distance between concepts can be derived. In particular, study to what extent concepts in graph theory can be represented by a lattice (or several). Deduce new concepts from this(these) lattice(s). □

Note that Graffiti is not designed for finding new concepts (except in the trivial sense that any inequality can be viewed as defining a new concept); it is claimed however in one place ([84]) that

> "*the current version can define its own properties.* One of the properties discovered by Graffiti is the class of all graphs in which the smallest eigenvalue has multiplicity 1. Graffiti defined this concept because it knew many examples of such graphs".

However, as no routine for concept discovery is described in the papers on Graffiti, this appears to be more an observation of the user than a discovery of the system.

PE4. Add to Graffiti a data mining routine to find frequent patterns in its database of graphs, as well as a routine and a database to check if they correspond or not to known concepts. □

Considering results of the heuristics, one may note that

(i) some of the conjectures of Graffiti which passed the tests are simple and attracted much attention of graph theorists;

(ii) the simplest ones are of the form $i_1 \leq i_2$, and the best known is probably conjecture Graffiti 2: *For any graph G*

$$\bar{l}(G) \leq \alpha(G),$$

(where $\bar{l}$ denotes average distance and α the independence number) proved by Chung [49].

It is surprising, as it connects very different concepts, on the one hand average distance, based on paths and on the other hand independence, based on non-adjacency. Perhaps this is the reason why it was not suggested by anyone before.

Other conjectures of the same form involve concepts which had been little studied, or not studied at all, by mathematicians at the time they were introduced into Graffiti; this is the case for the Randić index [144] defined for any graph $G = (V, E)$ by

$$Ra(G) = \sum_{i,j/v_i,v_j \in E} \frac{1}{\sqrt{d_i d_j}}$$

where d_j is the degree of vertex v_j. This concept appears in the following conjecture: *For any connected graph G*

$$\bar{l}(G) \leq Ra(G), \qquad \text{(Graffiti 3)}$$

which is still open (conjectures involving the Randić index tend to be hard to prove as the value of this invariant may increase or decrease upon addition of an edge to the graph considered).

Yet other conjectures use concepts invented by Fajtlowicz. The Havel-Hakimi operation on the set of degrees of vertices of a graph, ranked in order of non-increasing values, consist in deleting the first degree d_1 and reducing by 1 the next d_1 degrees. Havel [116] and Hakimi [103] independently proved that a degree sequence is *graphical*, i.e., corresponds to a graph, if and only if the degree sequence obtained by the above operation does. Iterating this operation finally leads to a series of zeros; their number is the *residue* $Re(G)$ Fajtlowicz considered it as an invariant and obtained with Graffiti the conjecture: *For any graph G*,

$$Re(G) \leq \alpha(G), \qquad \text{(Graffiti 69)}$$

which was proved by Favaron, Mahéo and Saclé [88]. Several further papers [66] [90] [96] followed.

The question of whether or not the concepts involved in a conjecture bear upon its interestingness has not been much studied. Fajtlowicz notes that finding new concepts is not difficult at all, contrary to the case of conjectures. Indeed concepts are not true or false, but simple or not, convenient or not and, more importantly, able or not to unify previous results. Finding new ones by computer is as easy as making guesses, but finding interesting ones may be another matter. Fajtlowicz argues that any sufficiently simple concept is interesting. While this may be true for most concepts which Fajtlowicz invented, as he found several nice ones (see Written on the Wall, *passim*), and attracted attention of mathematicians to them, it is hard to agree with his argument in general. Indeed, graph theory suffers from a plethora of concepts, the number of which suggests several questions.

First, to illustrate, one might argue that average distance is a simpler, or more central, concept than residue, and that the independence number is simpler and more central than both. Indeed, independence depends only on the basic concept of adjacency, average distance on the central concept of paths and their length while Residue depends on a particular algorithm. Of course, both average distance and residue give lower bounds on the independence number, and could be used in a branch-and-bound algorithm to determine its value. This may not be their main attraction, particularly for average distance which gives a usually loose bound.

Then considering general questions, we may propose:

RP5. Define the *simplicity* of a concept by the *minimal* number of operations to be applied to a graph G to compute it (operations not being considered here as elementary operations as in complexity theory but in more abstract terms as "checking adjacency for all pairs of vertices" or "computing all shortest distances between pairs of vertices"). This research would continue that of Graffiti on distance between concepts. □

RP6. Do the same as RP5 but using the concept of *Information (or Kolmogorov) Complexity*[127], i.e. the minimum length of a program to compute the invariant considered.

RP7. Evaluate empirically the importance of concepts in graph theory by a statistical analysis of their use in the literature. □

The next research proposal is inspired by the analysis of research networks as done in scientometrics [141] [126].

RP8. Construct a network of graph-theoretical concepts by associating them to the the vertices of a complete graph, and weighting edges by the number of times concepts corresponding to their end vertices are used in the same paper of some chosen corpus. Then analyze this network with standard tools of scientometrics to find central concepts, cliques of concepts used jointly, distance between concepts and other information. □

3.4 Refutation

Conjectures which passed the heuristic tests (or some of them) are tested on the database of graphs for correctness. If they do not hold for one of these graphs they are deleted.

Several remarks on the selection of graphs, heuristic or exact algorithms and graph representation are in order, as these questions bear upon the efficiency of the refutation process.

First, checking conjectures on the few hundred graphs of the database is not a severe test. Indeed, the classes of graphs under consideration are usually infinite.

Other systems are more powerful and/or more original in this respect: GRAPH uses interactive modifications, which constitute an informal descent method and can also get out of local optima; AGX applies the efficient and versatile Variable Neighborhood Search metaheuristic (see below); *Geng* and other enumeration programs list systematically much larger sets of graphs; INGRID combines relations between graph invariants, assuming the conjecture to be true, i.e., a *temporary theorem*, in order to derive a contradiction.

Some hybrids of Graffiti and enumeration programs have been sporadically explored: Fajtlowicz mentions using the CaGe program of Brinkmann [29] to generate fullerenes and De La Vina [68] applies Makeg of Mc Kay to obtain all trees satisfying given constraints and uses them in Graffiti.pc. She proposes as criterion of interestingness the *touch number* or number of graphs for which the conjecture is sharp (a criterion already used informally in [31] where it seems to have been mentioned in print, without the name, for the first time). In a recent paper, De La Vina [69] claims it was used in Graffiti since the early 90's, but for some reason it was not mentioned in the previous papers on that system, and notes that with a large database, conjectures with an important touch number tend to be true. Such a development appears to be promising.

Second, graphs in the Graffiti database are often those which refuted some conjecture and were proposed by various researchers. A set of 195 of them is described in the "*Graphs of Graffiti*" file [156]. The implicit assumption behind their selection appears to be that graphs which have refuted some conjecture may be more useful than randomly generated ones to refute others. This appears to be worthy of further study.

RP9. Study statistically which graphs are the most efficient for refuting conjectures of a given corpus, representative of the various types of algebraic ones. Examine also which conjectures are hardest to refute. □

Third, Graffiti (or Algernon) uses heuristics to compute the value of invariants such as the independence number, which are NP-complete to determine. This introduces an unnecessary error for small graphs; moreover up-to-date heuristics and metaheuristics could be used for evaluating such invariants, instead of simple heuristics such as MAXINE ([83]) for the independence number which are adequate for small graphs but not competitive for larger ones (see e.g. [14] [112] for state-of-the-art heuristics for the clique or independence number).

PE5. Replace heuristics for NP-hard invariants in Graffiti by exact algorithms, coupled with the best available heuristics for the same problems, to be used on large instances. □

In addition to automated refutation the process of finding conjectures with Graffiti uses further counter-examples obtained by hand by the user. In the DALMATIAN version, after introducing the corresponding graph(s) into the system a conjecture is generated again.

Here, knowledge and work of the user is incorporated and may strongly influence the quality of the conjectures obtained. Indeed, it is well known that when discovering and proving a theorem one often goes through a sequence of conjectures and refutations getting progressively closer to the correct statement. So this procedure is certainly reasonable and appears to be efficient; however, it is not automated. Probably, as discussed above, in the present state of graph theoretical theorem proving, it could not be. However, what is examined here is the impact of the counter-example obtained by hand on the new, further conjectures obtained. This is essential to evaluate how far the process of finding conjectures with Graffiti is automated.

To illustrate, consider Graffiti conjecture 117. Initially, this conjecture was stated as follows: *For any connected graph*

$$\bar{l}(G) \leq \sum_{j=1}^{n} \frac{1}{d_j}$$

It was disproved by Erdös, Pach and Spencer [77]. Fajtlowicz then proposed the weaker version:

For any connected graph G with girth $g(G) \geq 5$, average distance is not more than inverse degree (where inverse degree is shorthand for the sum of inverses of degrees of all vertices) and surmised that given the known counter-examples, Graffiti would come up with that version. Granting the hypothetical, it remains that a non-trivial result by famous graph theorists was needed to transform an initial conjecture which turned out to be false into an interesting and still open one.

Another example is Graffiti's conjectures 67 and 119; they involve the new invariant $f(G)$ defined as the *maximum frequency of occurrence of a degree in G* (or mode of the degree sequence). For conjecture 67, i.e.,

For any graph G without K_3 (i.e., *with* $g(G) \geq 4$)

$$\chi(G) \leq f(G)$$

counter-examples were found by Staton and later by Erdös and Staton [78]; knowing some counter-examples, conjecture 119 was obtained:

For any graph G without K_3 or K_4, (i.e., *with* $g(G) \geq 5$) $\chi(G) \leq f(G)$.

So, once again, a counter-example obtained by hand was needed to transform a false conjecture into an interesting open one. Recently, Caro [39] proved that this last conjecture is true for all sufficiently large graphs.

The fact that this step is not automated does not appear to be discussed in the parts of papers on Graffiti which concern automation. One may wonder how often one had recourse to counter-examples obtained by hand before reaching the conjectures publicized in "Written on the Wall". Very recently some information on that point has been provided in [80], it is stated that

"... in the 1980's once the conjectures were output, then as described by Fajtlowicz in [81] he would categorize the program's conjectures as *false*, *proven* and *open*. Counter-examples to conjectures were reported to the program, the program was re-executed and again the conjectures would be categorized. As further described in [80] after a few rounds of this process, as is the academic custom, Fajtlowicz announced the open conjectures".

3.5 Proofs

Many conjectures of Graffiti are true but trivial. Some of them are deleted as they are not informative according to the criterion of the DALMATIAN heuristic. This selection process could be made much more efficient by considering true relations (theorems) as well as conjectures.

PE6. Add to Graffiti a database of theorems containing both classical ones and others, proved with possible help of that or other systems. Then apply the DALMATIAN heuristic with a joint database of conjectures and theorems. □

True conjectures which pass the DALMATIAN heuristic test are studied by the user, and discarded if they appear to be trivial (which is not synonymous with, but implies the conclusion that they are trivial to prove). No operational system for theorem-proving in graph theory being available, this is done by hand.

Note that if a database of theorems is available it can also be used, as in INGRID [28], to find if a conjecture is implied by one or several theorems from that database and which. Then, if the resulting system is not too complicated, a proof might be obtained automatically by a system for algebraic manipulations such as Mathematica [161] or Matlab [130].

Presently, that one conjecture obtained with Graffiti (or a theorem if it has been proved) follows from another is only discovered with a web database or by a chance remark from one or another graph theorist.

To illustrate, the conjecture Graffiti 1 is: *For any graph G,*

$$\chi \leq 1 + rank(A(G))$$

where $A(G)$ is the adjacency matrix of G. Jaeger told Fajtlowicz ([79], p5) that Van Nuffelen [160] had proposed earlier the stronger conjecture

For any graph G,

$$\chi \leq rank(A(G)).$$

Both conjectures were refuted by Alon and Seymour [4].

This example shows the interest of a database of graph theory formulae, as discussed in Section 2.

3.6 Selection of conjectures

Until the version of Graffiti comprising the DALMATIAN heuristic, conjectures which passed the tests of the heuristics and could neither be refuted nor proved were further selected by the user. This new heuristic raised big hopes ([84], p 370):

> “There are strong indications that the new version of Graffiti can be used so that it will make very few trivial conjectures ... If these early indications, based on test runs, are right, it would mean that the program can be fully automated and can make conjectures without any help of humans. By contrast, as I was always clearly stating this, conjectures of previous versions of Graffiti had to be approved by myself, before they were included in “Written on the Wall”.”

However, it seems that proofs of easy true conjectures are still done by hand, perhaps some non-automated selection of conjectures still takes place and counter-examples obtained by hand are added to the database within the conjecture-making process. Automation of Graffiti is further discussed when considering the "Little Red Riding Hood" version of Graffiti in Subsection 3.8 below.

A few studies allow evaluation of the proportion of conjectures of "Written on the Wall" which are false. The two first of them correspond to the initial version. Favaron, Mahéo and Saclé [88] studied extensively eigenvalue properties of graphs conjectured with Graffiti. They proved 3 of them in their original form, 9 others as corrolaries of stronger results and disproved 49 of them. Brewster, Dineen and Faber [24] program a series of invariants and tested about 200 conjectures of Graffiti using a database of all graphs with up to 10 vertices. They refuted 49 of these conjectures (some with such simple graphs as a single edge and proved one).

As the DALMATIAN heuristic is more selective than previous ones, one may wonder if conjectures obtained with the DALMATIAN version of Graffiti are more often true than before. They are numbered from 700 upwards in "Written on the Wall". Pujol [142] studied 12 conjectures, in that range pertaining to cubic graphs. For that purpose he used the AutoGraphiX system (see Section 4 below) in interactive mode together with a program for cubic graph enumeration, due to Brinkmann [29]. 5 out of the 12 conjectures could be refuted. For the other ones, it was shown that a minimal counter-example would have at least 18 vertices. While this is a small sample, it nevertheless indicates that the proportion of false conjectures obtained with the DALMATIAN version of Graffiti, and after elimination of false or trivially true conjectures by both automated and non-automated methods may still be large.

3.7 Minuteman and Discriminant Analysis

The Minuteman version of Graffiti [86] is designed to solve problems of discriminant analysis, i.e., separating entities from given sets by values of a function, which corresponds geometrically to a surface, often a hyperplane. A motivating application was to discover stability sorting patterns of fullerenes. An additional routine works as follows ([85] p.21):

> "To study conjectures, objects are sorted by the difference between both sides of the inequality and sometimes when this is done for fullerene conjectures they show a conspicuous pattern by displaying the known stable examples on the top of the list and those with the largest sum of eigenvalues (i.e., presumed candidates for the least stable) at the bottom".

We do not discuss the chemical relevance of the patterns and conjectures so found here. Regarding the routine, note that checking if there is a pattern in the one-dimensional data obtained for a conjecture is done visually. It could of course easily be automated and simple statistical tests applied.

Now, if computer-assisted or automated systems for conjecture-making in graph theory are still rare, the situation is completely different in discriminant analysis. Indeed, this is a well established field, beginning in statistics at least 65 years ago [91] and presently central to data mining. Automated methods to find separating planes or surfaces in low or high dimensional spaces are operational for various criteria. Let us just mention that if perfect separation by a plane is possible this can be done by linear programming [129] and that otherwise one can use *decision trees* [143] [23], *support vector machines* [46], *logical analysis of data* (LAD, [19]) or other methods.

So while Minuteman is far from the state of the art in discriminant analysis, it suggests the interest of using more powerful discrimination methods in graph theory. In a similar vein, Colton [52] recently stressed that mathematics could be viewed as a new field for data mining. Some techniques using Boolean variables appear particularly well-suited to the case of graph problems, e.g. LAD and decision trees.

RP10. Apply decision trees and LAD to discriminant problems in graph theory. Such problems may be mathematical ones (e.g. belonging or not to a particular class of graphs) or applications based on measurements relative to the problem under study (as in the fullerene example discussed above). Compare results with those of other conjecture-making systems. □

RP11. Study criteria for approximate separation in graph theory using various discriminant analysis methods, both for mathematical problems and for applications. Examine when and how an approximate separation (e.g. a linear one) can lead to an exact one (e.g. by restricting the class of graphs considered or barring exceptional cases). □

3.8 Pedagogical versions of Graffiti

Computer systems have long been used with success in teaching graph theory. This was already the case of GRAPH [59], Chinn [41] reports on her use of INGRID for that purpose and the "CABRI-graphes" system [38] developed in Grenoble led to the widely distributed "CABRI-géomètre" package.

Recently, versions of Graffiti devoted to teaching graph theory with an active pedagogy were developed. They met with equal success when used in special project classes. Pepper [137] gives an enthusiastic record of his discovery and use of Graffiti, and Chervenka [40] describes more briefly how she used De La Vina's Graffiti.pc [68].

The main difference with previous versions of Graffiti is in use: initially the database of graphs is empty. When a first graph is entered, conjectures are formulated and the corresponding invariants studied. These conjectures are often easy to prove or refute. For that reason, more work is asked from the students than merely to provide a counter-example: they are requested

(i) if the conjecture is refuted, to find a smallest counter-example in terms of number of vertices and, as a secondary criterion, of number of edges;

(ii) if the conjecture is true, to determine whether it is NP-hard or not to determine if a graph G satisfies the relation (assumed to be an inequality) as an equality.

While such tasks are initially easy to accomplish, their difficulty will augment with the number of graphs in the database. Graffiti does not contain routines to do them automatically or in computer-aided mode. At an early stage, this is reasonable if one wants the students to practice their refuting and proving skills. Later, they might want to have some help, which could be provided by a system such as GRAPH or AGX.

PE7. Add to the pedagogical versions of Graffiti a program for visualizing graphs on screen, modifying them online and computing automatically a series of invariants or formulae involving invariants. □

While the main advantage of such an enhancement would be to make interaction with the system easier and more effective, another one would be to show students that tedious computations may be delegated to the machine, so that they may concentrate on reasoning.

PE8. Add to the pedagogical versions of Graffiti a routine similar to AGX's function for evaluating invariants subject to constraints: then use it if the relation is false to find smallest counter-examples by parametrizing on numbers of vertices and edges and attempting to find a graph which does not satisfy the given relation. □

Such an enhancement should be made available only after students have tried to find minimum counter-examples on their own, and submitted them to the system.

PE9. With the same function, and assuming that the relation is true, find graphs which satisfy it as an equality, to help estimate how difficult it is to recognize them. □

The same comment as for PE8 holds here too.

In the "Little Red Riding Hood" version, the task of proving true conjectures is ignored. It is then claimed ([85] p. 18) that it is "an offshot aimed at fully automating the program apart from the invention of concepts".

Observe however that no additional functions have been automated since the last general version. Some tasks have been abandoned (proving true conjectures) and some others, done by hand, made harder (finding counter-examples which are smallest possible). As previously, the fact that generation of counter-examples is not automated is overlooked in the comment cited above. In fact, steps of automated generation of conjectures alternate with steps of finding smallest counter-examples, in what appears to be a typically interactive man-machine process.

3.9 Complexity and the $P = NP$ problem

Some considerations on the condition for stopping of "Red Burton" and its relation to complexity issues, i.e., the $P = NP$ problem, are given in ([85] p.24). As many researchers (e.g. Smale [158]) consider this last problem as the most important one of computer science we examine this text in detail.

A first paragraph tells us that:

> "Once in a while it may happen that all conjectures of a given round are true. The natural interpretation of this situation-called *bingo*- is that for every object (under consideration, not just those in the database of the program) there is

> a conjecture made in this round such that the left and the right sides of the inequality have the same value for this object. Unless this indeed is the case, supplying the program with a counter-example to this situation will still break the stalemate and one can proceed to the next round."

One may wonder if this interpretation is "natural"; the set of objects (graphs) under consideration may be very large, and in some cases, discussed below, infinite. Extrapolating the fact that there is a tight relation for every object in the database to this much larger set is a very risk step. But, clearly, as indicated, if this property does not hold and one can find an object for which there is no tight relation, one can proceed to the next round. Note that this task is different from those of Red Burton as described earlier in [85], in which one asks for objects which *refute* a conjecture, not for objects for which no conjecture is *tight*.

The next paragraph of the text begins as follows

> "Most of the interesting runs of the program will yield at least one false conjecture in each round. This will always happen if the leading invariant L is NP-hard and all the remaining invariants from N are polynomially computable. These versions of the program will run forever modifying some of its conjectures after each round. Some of the conjectures are cyclically and some are continuously repeated in rounds providing more and more experimental evidence for their correctness."

The second sentence does not appear to be true. No proof is given and a counter-example is easy to find: let the leading invariant L be the independence number α, the class of graphs under consideration being all non-trivial graphs, i.e., all graphs with at least one edge and the only graph in the database in the first round a star, say S_4. Then the system will give the relation $\alpha(G) \leq n-1$ and the first round will end without a false conjecture. If another graph, say C_4, is introduced, there will be an infinite, incomplete second round, still without a false conjecture. Moreover, if as stated at the beginning of the third sentence

> "These versions of the program will run forever ...",

it does not seem they can lead to a "bingo" which implies that they stop. This contradicts the first statement of the remainder of this second paragraph, next reproduced, which gets to the main question:

> "One can still end up with a correct *bingo* but this would imply $P = NP$ in which case the more appropriate term for the situation would be "big bang". Penrose does not question that in a sense a machine's insights may be superior to human. It is not unthinkable that $P = NP$ can be proved, because machines may conjure up hundred of novel radius, average distance, residue, and δ-like bounds, constituting a valid bingo."

For the last sentence to make sense, one should write "big bang" instead of "bingo". Then, one may wonder if it is true, and if it has information content. Recall from elementary logic that B holds because of A is equivalent to the implication $A => B$ and means that A is false or B is true. Let B denote the proposition "It is not unthinkable that $P = NP$". As long as it has not been proved that $P \neq NP$ this is a tautology, i.e., certainly true. But then the implication holds regardless of the antecedent A, i.e., one can adopt for A any statement whatsoever, true or false, instead of "machines may conjure...". So for the last sentence to have information content, one must show that a "big bang" has some *plausibility* not that it is merely *possible*. For this to be done along the proposed lines one must show it is plausible that one can:

(a) find relations given sets of graphs (various systems do this);

(b) find in each round graphs which are not tight for any of the relations involving the chosen invariant and direction in the database of conjectures. This task increases in difficulty with the size of that database. Moreover, such graphs are likely to be increasingly and finally enormously large (clearly no machine could find such graphs if they must have billions of vertices).

(c) prove that *all* relations considered in the last round are true;

(d) prove that there exists *no* graph under consideration, the set of which is necessarily *infinite* for the problem under study to bear upon $P = NP$, for which none of the relations considered in the last round are tight.

Clearly, this proof scheme is incredibly difficult to carry out. Except for step (a) the necessary steps are not even listed, nor of course discussed. As no argument is provided for a "big bang" to be plausible, the last sentence of the cited text has no information content. In other words, that "machines may conjure up hundred of novel ... bounds" provides no argument of any weight for or against $P \neq NP$.

4 AutoGraphiX (AGX)

4.1 Uses and structure

As mentioned in the introduction AGX has several aims. We focus here on computer-assisted and automated conjecture-making. Indeed, AGX can be used in both modes, and the steps involved as well as their sequence must be carefully distinguished.

When working in computer-assisted mode, AGX's follows the following ones.

Step 1. Problem formulation.

Step 2. Obtention of a set extremal or near-extremal graphs for the chosen objective subject to the stated constraints.

Step 3. Visual display of the graphs found and parametric value curves.

Step 4. Interactive improvement of graphs which do not appear to be optimal.

Step 5. Interactive derivation of structural and algebraic conjectures.

When AGX is used in automated mode, steps 3 to 5 are replaced by the following ones

Step 6. Recognition of extremal graphs belonging to known families.

Step 7. Determination of linear equations between invariants associated with all or some subset of the external graphs obtained by the numerical method.

Step 8. Determination of linear inequality relations between invariants by the geometric method.

Step 9. Determination of linear or nonlinear relations between invariants by the algebraic method.

Step 10. Results: output external graphs found, families to which they belong, parametric curves of values for the objective, and conjectures found.

Note that not all methods for finding conjectures automatically need be used in the same experiment: one of steps 7, 8 or 9 suffices; step 6 is also optional except if step 9 is used.

4.2 Problem formulation

When the aim of using AGX is conjecture-making, one leading invariant is usually selected and others (most often n and m) used as parameters. Moreover, the class of graphs considered is specified by constraints, which will be added to the objective function with large coefficients (as in Lagrangian relaxation). Such coefficients must be chosen to be sufficiently large to exclude any graph not in the class considered; if this is not possible, a large value indicating a contradiction will be obtained.

Moreover, in some cases it is necessary to add a secondary criterion or progressive series of weights in order to transform the graphs in directions which will tend to satisfy the constraints.

An example occurred at Graph Theory Day 42, where after a presentation on *Computers in Graph Theory* [104] a demonstration of AGX was made. Cowen [53] asked for graphs with a maximum number of K_4 for a given number of K_3. This last number was chosen as a parameter and the number of K_4 maximized, which led on the spot to rediscovery of a series of extremal graphs for those parameters. Running AGX for a longer time gave a series of further extremal graphs of larger size.

At another (early) demonstration of the system, Seymour [151] asked for cubic graphs of diameter 3 with a maximum number of vertices. A first try where the diameter was minimized under the constraint that all degrees be equal to 3 yielded examples with 14 and 16 vertices but not more (the constraint on the degree was imposed by penalizing the numbers of vertices of degree smaller or greater than 3 increasingly with their distance to that value). Adding as secondary criterion minimization of the average distance, and so smoothing the objective function, led to cubic graphs of diameter 3 with 18 and 20 vertices in 35 seconds and 1 minute respectively; the latter graph is optimal.

Presently AGX disposes of about 60 invariants to be used in the objective function and constraints. They are *order*, *size*, *independence number*, *chromatic number*, *chromatic index*, *minimum degree*, *maximum degree*, *average distance*, *degrees of the vertices*, *eigenvalues of the adjacency matrix* and others.

However, this is not a very large set, as compared with those of GRAPH, Graffiti, LEDA or other systems.

PE10. Add to AGX routines to compute the main graph theoretic invariants not yet included (e.g. *matching* number, *domination* number, etc) □

Graph invariants are invented every day, so if AGX is to accommodate all needs of the users, it must let them add their own routines for their favorite invariants.

PE11. Construct a version of AGX in which the user can add routines to compute new invariants. □

This enhancement is being implemented in the new version, AGX2, of AGX, which is currently being built.

At present, standard algebraic expressions can be taken in the objective function and constraints as well as some simple graph transformations such as complementation. Other operations should be made possible.

PE12. Add to AGX routines for the main graph operations, such as sum or product of graph, etc, as done in GRAPH.

4.3 Finding extremal graphs

The principle of *AGX* is to use heuristic optimization to find a family of extremal or near-extremal graphs for some objective, subject to constraints, then to exploit the corresponding information.

Heuristic optimization in *AGX* follows the Variable Neighborhood Search (VNS) metaheuristic [108], or framework for building heuristics. VNS exploits the still rather new idea of systematic change of neighborhood within the search. This is done in two ways: first in a descent routine, called Variable Neighborhood Descent (VND), which leads to a local optimum, and, second, in a systematic effort to get away from this local optimum by applying increasingly strong perturbations and descents.

Rules of VNS are as follows:

0. Select the set of neighborhood structures $N_k, k = 1, \ldots, k_{\max}$ that will be used in the search for a better local optimum, and a stopping condition. Find an initial solution (or graph) x.

Repeat until the stopping condition is met:

1. Set $k = 1$;
2. Until $k = k_{\max}$, repeat the following steps
   - (a) (*shaking*) generate a point x' at random from the k^{th} neighborhood of x (i.e., $x' \in N_k(x)$):
   - (b) (*descent*) Apply the Variable Neighborhood Descent routine with x' as initial solution: denote by x'' the local optimum obtained;
   - (c) (*improvement or continuation*) If the solution x'' so obtained is better than the best known one x, move there ($x \leftarrow x''$) and continue the search within $N_1(x)(k = 1)$; otherwise set $k \leftarrow k + 1$.

The stopping condition may be a maximum number of iterations, a maximum CPU time or a maximum number of iterations or CPU time since the last improvement.

Rules of VND are as follows:

0. Select the set of neighborhood structures $N'_k, k = 1, 2, \dots, k'_{max}$ that will be used in the descent. Consider an initial solution x.

Main step: Set $k = 1$ and $i = FALSE$ (*improvement indicator*).

Until $k = k'_{max}$, *repeat the following steps:*

(a) Find the best neighbor x' of x in $N'_k(x)$;

(b) If the solution x' so obtained is better than x, set $x \leftarrow x'$ and $i = TRUE$;

(c) Set $k \leftarrow k + 1$;

(d) if $k = k'_{max}$ and $i = TRUE$ set $k = 1$.

In words, VND applies a series of transformations to the current graph, keeping each time that transformation giving the best improvement. It there is no improvement within the current neighborhood, VND proceeds to the next one. If there is no further improvement when considering all neighborhoods in turn, VND stops; otherwise it begins again at the first neighborhood.

Moves corresponding to the different VND neighborhoods in AGX are the following: *rotation* of an edge, *deletion* of an edge, *addition* of an edge, *move* of an edge, i.e., deletion plus addition, *detour*, i.e., removal of an edge and addition of two edges between endpoints of the deleted one and a vertex not adjacent to either of their endpoints, *short cut*, i.e., the operation that is the reverse of detour, 2-*opt*, i.e., removal of two non adjacent edges, and addition of two different edges connecting the endpoints of the removed ones: *add pendant vertex*: i.e., add a new edge from an old vertex to a new one; *delete vertex* of bounded degree and all adjacent edges.

The neighborhoods rotation, addition, deletion and move are the most frequently used, and the least time consuming ones.

If all moves within a neighborhood are examined before choosing the last one, only graphs of moderate size may be considered, particularly if the objective function to be computed after each potential move is hard to evaluate. A speed-up can be obtained by using a "first improvement" instead of a "best improvement" rule.

The choice of moves is presently left to the user (with a standard option of using them all). However, this choice could be automated:

PE.13 Add a routine which evaluates the effect of all moves during an initial period, then selects for continuation of the search those which proved to be the most efficient. □

One could also try to find new moves systematically:

PE.14 Construct moves by all possible transformations on a small graphs (e.g. with 4 vertices). Eliminate redundant ones which are the same as others up to symmetry. □

Contrary to VND, which uses systematically a series of different local moves, VNS makes random use of more global moves, often deriving from a simple principle. The most frequently used one is to repeat a move k times, e.g., one first moves an edge chosen at random (a move in $N_1(x)$) then 2 (a move in $N_2(x)$) and so on.

VNS appears to be quite powerful, i.e., very often, but not always, it gives extremal graphs. Cases where it does not give the best graph, which can often be recognized by comparison with graphs obtained for close values of the parameters, can be exploited to define new neighborhoods.

RP12. Systematically explore cases in which the neighborhoods of VND and VNS are not enough to find consistently extremal graphs. Define new neighborhoods accordingly and study the complexity of implementing them.

4.4 Display of results

Results of *AGX*, when used in interactive mode are of two types:

a) Extremal or near-extremal graphs;

b) Parametric curves of values of the objective function.

Extremal graphs can be visualized on screen or printed. Drawings can be modified interactively by moving vertices; classes of vertices or of edges can also be highlighted, in various colors (e.g. edges which are critical for some invariants, edges of a spanning tree or a shortest path tree, vertices of various degrees, . . .)

Up to now only simple tools of graph drawing have been implemented in *AGX*, i.e., a specialized routine for representation of trees with edges parallel to the axes, a "spring" type heuristic to avoid cluttering parts of the drawing with closely spaced vertices, and a few more. As the field of graph drawing is very active (see e.g. Di Battista et *et al.* [70] [71]) further results obtained there could be exploited. Note however that the frequently adopted criterion of minimizing edge crossings does not seem adequate for AGX's needs. Easy recognition of subgraphs of one or another type (*cliques*, *cycles*, . . .) seems more important.

RP13. Study precise needs of AGX for graph drawing and how they can be met by methods of that field. In particular consider ways to make structure (particular subgraphs, graphs formed from them) visible in individual graphs as well as in sequences of graphs.□

While there may be no closed-form formulae for some invariants on general graphs, there may be some for particular classes of graphs. Recognizing them can lead to conjectures, as discussed further below.

Curves of values can be represented in three dimensions, corresponding usually to some invariant i_1, n, and m. These curves can be rotated, superposed, isolated, etc. . . Moreover, graphs corresponding to particular points on these curves e.g. minima or maxima can be displayed in a window. Finally, if one has some idea about a conjecture it can be introduced into AGX, checked, displayed with the curves, and both the differences in ordinates and the points of contact highlighted.

These facilities should be extended to higher dimensions.

PE15. Add to *AGX* a routine for projection of points in R^p with $p > 3$ but moderate, corresponding to extremal graphs, on subspaces with 2 or 3 dimensions. □

4.5 Recognizing structure interactively

When visualizing the extremal graphs obtained with AGX, it is not uncommon to find that

(i) they belong to some well-known family, e.g. paths, circuits, trees, stars, bipartite, complete,. . . ,

or

(ii) they have some recognizable but more complicated structure.

There may be some exceptions among them, and one should then find out whether this is due to the VNS heuristic not finding the (or an) extremal graph for the corresponding values of the parameters or to the particularities of the objective function under study.

Usually, significant differences in structure or an outlier position with respect to the curve of values make such exceptions conspicuous. One can then deduce from close examples what might be the true extremal graph for those parameter values and build it by moving edges with the mouse; then the system will compute its value and, if it is better than the previous near-extremal graph, substitute them.

This step is not mandatory, and not used when applying AGX in automated mode (outliers may be removed in other ways, see below). However, it could be automated, or at least more automated than it presently is.

PE16. Augment the number of routines for recognition of classes of graphs in AGX.□

PE17. Add a routine which will test if extremal graphs frequently belong to some parameterized family; for those parameter values for which it is not the case, compute their value and substitute them if there is an improvement. □

Note that such developments are close to those needed in the third (algebraic) way to find conjectures automatically (see below). The automated parts correspond to step 6 of AGX, which will not be discussed further.

4.6 Obtaining conjectures interactively

Conjectures most often made have the two following forms (see also [113])

(i) *Algebraic relations* between graph invariants, valid for some class of graphs (e.g. all graphs, connected, bipartite, split, stars, trees, complete. . .)

(ii) Description of the *structure* of extremal graphs (i.e., of a known or new class of graphs) or of a subset of them.

Conjectures of the former type can be obtained from the parametric curves of values of the objective. Consider for instance the *energy* E of a graph defined [31] [97] [98] as

$$E = \sum_{i=1}^{n} |\lambda_i|.$$

Minimizing this function with parameters n and m, then superposing the curves of $E(m)$ for fixed n shows very clearly all values to be above a parabola. Its equation is then readily found and leads to the lower bound [31]

$$E \geq 2\sqrt{m}.$$

Moreover, the equation of this curve can be entered in AGX, which represents it in the plane of values and highlights points where it is attained, i.e., graphs reaching the bound, as well as differences for other points. One can thus see if the bound is sharp and if it remains so over the range of parameter values or not. In the case discussed, when m becomes large the curve lies increasingly below observed values. Then, looking at curves for one value of n at a time suggests a linear lower bound for each, from where the inequality

$$E \geq \frac{4m}{n}$$

follows. It is sharp for fewer values than the first bound, though still sharp several times.

Conjectures of the second type are obtained by examining the graphs obtained and, possibly, exploiting conjectures on these graphs obtained automatically (see below). Sometimes, results are straightforward. For instance minimizing with AGX the energy of unicyclic graphs (a problem of interest to chemists) led to extremal graphs which were cycles for $n \leq 7$ or n =9, 10, 11, 13 and 15 and 6-cycles with an appended path for all other values of n considered. The natural conjecture that these and only these graphs were the true extremal ones [31] has recently been partially proved [99] [117].

4.7 Numerical method of conjecture-making

We now turn to the automated mode of using AGX and consider the three ways in which this has been done (up to now). A first method uses the mathematics of principal component analysis to find resemblances between objects, in the form of affine relations they all satisfy, instead of differences as usually done.

The method works as follows [35] [36]:

(a) Find extremal or near-extremal graphs for some objective with AGX;

(b) Filter this set to remove outliers (optional but often useful);

(c) Compute values for a set of invariants on all remaining graphs;

(d) Center the vectors of values for each invariant (thus transforming the problems of finding affine relations into that of finding linear ones);

(e) Compute the variance-covariance matrix V between centered vectors;

(f) Diagonalize V, with, however, some empty lines if there are relations. In the resulting matrix V', $Dim(I_m(V))$ lines contain non-zero terms and correspond to independent variables. The remaining $n - Dim(I_m(V))$ lines contain only zeros and correspond to dependent variables which may be expressed as linear combinations of the independent ones. These relations form a basis of the null-space of V. Using the initial data one can then compute the right-hand sides of the corresponding affine relations.

To illustrate, consider the irregularity $irr(G)$ of a graph G as defined by Albertson [3]: let the *imbalance* imb_{ij} of edge (v_i, v_j) of G be defined by

$$imb_{ij} = |d_i - d_j|$$

and the irregularity $irr(G)$ of (G) by

$$irr(G) = \sum_{i,j|(v_i,v_j)\in E} imb_{ij}.$$

Applying AGX [107] led automatically to the following conjectures valid for graphs G with maximum irregularity:

$$\begin{aligned} r(G) &= 1 \\ \chi(G) &= \omega(G) \\ n &= \Delta + 1 \\ \alpha(G) &= -\omega(G) + \Delta + 2, \end{aligned}$$

from where it follows that

$$\alpha(G) + \omega(G) = n + 1$$

which implies that the extremal graphs are split graphs, i.e., graphs consisting of a clique, a disjoint independent set and edges joining vertices of the clique to those of the independent set. (For further use of this information and of the external graphs found, see [107]).

Several comments are in order. First, note that the algorithm described takes polynomial time: if the number of graphs considered is fixed and t invariants are computed, it requires $O(t^3)$ time.

Second, observe that it gives relations for *subsets* of the set of invariants considered, not necessarily for the whole set. So the combinatorial problem of finding the right subset is avoided. This implies that given a sufficiently large set of graphs of some class, and sufficient computing time, one could find a basis of affine relations among a large set of invariants (maybe several hundred of them). Should such relations exist and be up to now unnoticed, it would prove that interesting relations may be found without focussing on a particular problem, or domain (such as e.g. problems of distances in graphs).

Third, the algorithm subsumes some other ones, which have met with success. For instance the BACON algorithm developed by Simon and co-workers [122], gives *rational reconstructions* (or possible reasonings) for great discoveries of the past in physics and chemistry. It uses four rules, given a set of observations involving several variables:

(a) If a variable is constant, a law has been found;

(b) If a variable is a linear function of another one, a law has been found;

(c) If a variable increases while another one decreases, add a new variable equal to their product, and iterate;

(d) If a variable increases while another one increases, add a new variable equal to their ratio and iterate.

BACON rediscovered Kepler's third law, in three iterations only, as well as several other famous ones. The numerical method of AGX reproduced these results in much less computer time [36]. These laws are expressed as monomials, i.e., products of variables with integer powers. Taking logarithms gives affine functions.

However, there are many more complicated cases: Langley *et al.* [122] have observed that laws in chemistry may take a more general form, the logic of which had, apparently, not yet been studied [155]: in addition to the variables, there are substance-specific constants, such as e.g. *specific heat.* BACON could be extended to this case.

RP14. Study how to extend the numerical method of AGX in order to apply it to problems with both variables and substance-specific constants. □

Fourth, one would clearly like to extend the discovery of conjectures to more general cases than affine relations. Note first that inequalities are obtained in a straightforward way: it suffices to check on which side lie graphs which are not extremal for the objective under study. Then one might add, as new variables, products of variables, or simple powers such as squares, cubes, inverses, square roots and the like.
All this increases the number of variables, and thus augments the number of graphs needed to obtain relations, as well as computing time, but does not change the method itself.

If e.g. only products of two variables are considered it is still possible to consider a few tens of variables. One would like to do better than this brute-force approach, and in view of results obtained by *support-vector machines*, this seems to be possible.

RP15. Study selection of product and power terms in finding nonlinear conjectures between invariants in graph theory. Devise corresponding heuristics. □

Fifth, even more general sets of relations involving *signomial functions* (polynomials in several variables with arbitrary powers and signs) have been studied by using neural networks. These have reconstructed with fairly good precision a set of such equations.

RP16. Compare conjecture-making by the numerical method of AGX and by neural networks; define hybrids where neural networks are used to find the form of the relations and AGX to find the precise values of coefficients. □

Sixth, to determine affine relations, which are equalities, numerical precision is required, and hence control of errors. Standard tools of numerical analysis are used to do so, but the guarantee of finding all affine relations is not complete. To attain such a goal one would need computations in error-free arithmetic (i.e., making computations with rational numbers using a sufficient number of digits to avoid all approximation errors), which has been used in solution of equations associated with Euler sums [12], but are very time consuming.

4.8 Geometric method of conjecture-making

Consider a set of extremal graphs for some objective; they correspond to points in the $\mathbb{R}^p$ space of invariants (or in a sub-space of selected invariants), each of which is associated with one of the p axes. Then constructing the convex hull of these points with a gift-wrapping algorithm (as e.g. implemented in the package of Avis and Fukuda [10]) immediately yields a set of conjectures in the form of linear inequalities: for each invariant, faces passing below all points, or above all points correspond to lower and upper bounds.

To illustrate, consider chemical graphs, in which $\Delta \leq 4$ due to the valency of carbon. The geometric method of AGX could find the two following relations in a very small computing time:

$$Ra(G) \geq \frac{n}{3} + \frac{m}{12}$$

and

$$Ra(G) \geq \frac{1}{4}(m + n_1).$$

The main difficulty with this approach is to avoid undue extrapolation. In the case of a function of a single invariant say $i_1(n)$, if it is concave, just the next graph could disprove the conjecture.

Therefore the conjectures obtained are systematically tested by looking for the few extremal graph(s) following those used to find them. Also the *touch number* criterion discussed above is of interest here: if the inequality found is sharp at only a couple of points, it appears to be of little interest. Conversely, if it is sharp for many or even most values of the parameters, as is the case for the two relations just cited, it is clearly interesting.

One would then like to obtain often sharp relations even when the relationships between invariants of extremal graphs are nonlinear. Again this could be done by introducing new variables, i.e., going to a higher dimensions. There are some limitations here, as gift-wrapping algorithms may become very time consuming with only 10 variables or so.

RP17. Examine how to transform functions in order to get nonlinear relations through the geometric method. Compare results with those of the numerical approach for the same problems.

4.9 Algebraic method for conjecture-making

The principle of the third method is to recognize extremal graphs for some objective function, then to use relations between invariants valid for those classes of graphs in order to obtain new relations, which are conjectured to hold in general.

To illustrate, consider the objective function $Ra(G) - \bar{l}(G)$, (which corresponds to conjecture Graffiti 3, i.e., $\bar{l}(G) \leq Ra(G)$). Minimizing this relation systematically gave stars, for which the Randić index is equal to $\sqrt{n-1}$ (and is minimum for fixed n as shown by Bollobas and Erdös [18]) and the average distance is $2 - \frac{2}{n}$. This leads to the conjecture
For any connected graph G

$$Ra(G) - \bar{l}(G) \geq \sqrt{n-1} + \frac{2}{n} - 2,$$

which strengthens Graffiti 3. If true, the new bound is sharp for all $n \geq 1$.

The difficulty of this method is the large amount of information needed: on the one hand, specific algorithms are required to recognize to which class belong extremal graphs, and on the other hand a database of relations between graph invariants is needed for each class considered. Presently this method is working in experimental mode.

PE18. Extend the set of graph recognition routines of AGX. □

PE19. Extend the database of relations between graph invariants. □

PE20. Couple the algebraic method with Mathematica or Matlab to simplify the relations obtained. □

Clearly it will not always be the case that extremal graphs all belong to a single well-defined class, for which relations are known. A first difficulty is then that one will have to use lower or upper bounds (e.g. if all one can find is that extremal graphs are trees), although that would not change the approach too much.

PE21. Extend the algebraic approach to manipulate bounds rather than equalities between invariants. □

Another extension would be to recognize the various classes of graphs which are extremal for some values of the parameters (a problem already evoked above) and modify again the way bounds are computed and relations obtained.

Finally, once again one should compare methods.

RP18. Compare systematically results of the three methods proposed for automated conjecture-making on the same set of problems, including some which led to well-known graph theorems. Deduce from this comparison intimations about what makes a relation difficult to find for one or all of them. □

5 Conclusions

Computer-assisted and automated conjecture-making in graph-theory appears to be very successful and has led collectively to more than 200 papers research reports and theses. This makes it probably the most active subfield of discovery science.

Three systems are operational and largely used: GRAPH, Graffiti and AGX. Their principles are different: interactive computing, generation of a priori conjectures and selection amongst them, heuristic optimization to get extremal graphs and deduction of conjectures from them. All three have large parts which are automated, but only the last can presently be used in (fully) automated mode, that is with a problem statement unaccompanied by further information, no human intervention between problem statement and reading the final results as well as no selection among results so obtained. Note that this is not the only way to use this system, nor necessarily the most efficient one, as interactive modification of the extremal graphs obtained may give insight on how to prove the conjectures it delivers.

All three systems (and others) are susceptible of fuller automation in the near future. A series of suggestions on 21 possible enhancements are given in this paper, as well as a list of 18 more general questions, or research paths, of possible interest to the whole field.

As a final point, observe that the conjectures considered in this paper are mainly algebraic inequalities (or, in some rare case, equalities) among graph invariants. As discussed more fully in [113] there are many other forms which interesting conjectures in graph theory can take. So there is plenty of room for further achievement in this young and promising field.

Acknowledgments:

This paper was written in part during a visit to SMG, University of Brussels; support of the *Research in Brussels* program is gratefully acknowledged as well as NSERC grant # 105574-98. Thanks to Mustapha Aouchiche, Gilles Caporossi, Hadrien Mélot and Dragan Stevanović for discussions as well as Dragos Cvetković and Siemion Fajtlowicz for correspondence which helped to clarify issues discussed.

References

[1] ABELEDO, H., and ATKINSON, G.W. The Clar and Fries problems for benzenoid hydrocarbons are linear programs. In: *Discrete Mathematical Chemistry*, P. Hansen, P. Fowler, and M. Zheng, Eds., vol. 51 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, Providence RI, 2000, pp. 1–8.

[2] ABELEDO, H., and ATKINSON, G.W. Polyhedral combinatorics of benzenoid problems. Proceedings of IPCO VI, Houston (1998). *Lecture Notes in Computer Science*, New-York, Springer 1412.

[3] ALBERTSON, M.O. The irregularity of a graph. *Ars Combinatoria*, 46 (1997) 215–225.

[4] ALON, N. and SEYMOUR, P. A counter-example to the rank-coloring conjecture. *Journal of Graph Theory*, 13 (1989) 523–525.

[5] AOUCHICHE, M. www.gerad.ca/AGX. A Bibliography on AutoGraphiX, its Results and Related Topics (forthcoming).

[6] AOUCHICHE, M., CAPOROSSI, G., and HANSEN, P. Variable neighborhood search for extremal graphs 8. Variations on Graffiti 105. *Congr. Numer.*, 148 (2001) 129–144.

[7] APPEL, K., and HAKEN, W. Every planar map is four colorable. Part I. Discharging. *Illinois J. Math.*, 21 (1977) 429–490.

[8] APPEL, K., and HAKEN, W. Every planar map is four colorable. Part II. Reducibility. *Illinois J. Math.*, 21 (1977) 491–567.

[9] APPEL, K., and HAKEN, W. Every planar map is four colorable. *Contemp. Math.*, 98 (1989) 1–743.

[10] AVIS, P., and FUKUDA, K. *lrs home page*; *cdd and ccd plus page.*

[11] BAILEY, D. Integer Relation Detection. *Computing in Science and Engineering* 2 (2000) 24–28.

[12] BAILEY, D.H., BORWEIN, P.B. and PLOUFFE, S.A. New formulas for picking up pieces of Pi. *Science News*, 148 (1995) 279.

[13] BAILEY, D.H., BORWEIN, P.B. and PLOUFFE, S.A. On the rapid computation of various polylogarithmic constants. *Mathematics of Computation*, 66 (1997) 903–913.

[14] BATTITI, R., and PROTASI, M. Reactive local search for the maximum clique problem. *Algorithmica* 29 (2001) 610–637.

[15] BEEZER, R.A., RIEGSECKER, J. and SMITH, B.A. Using minimum degree to bound average distance. *Discrete Mathematics*, 226 (2001) 365–377.

[16] BERGE, C. Färbung von Graphen deren sämtliche bzw. deren ungerade Kreise starr sind (Zusammenfassung), *Wissenschaftliche Zeitschrift, Martin-Luther-Universität Halle-Wittenberg, Mathematisch-Naturwissenschaftliche Reihe*, (1961) 114–115.

[17] BERGE, C. Perfect graphs I. *Six papers on graph theory.* Indian Statistical Institute, Calcutta (1963).

[18] BOLLOBAS, B. and ERDÖS, P. Graphs of extremal Weights. *Ars combinatoria*, 50 (1998) 255–233.

[19] BOROS, E., HAMMER, P.L., IBARAKI, T., MAYORAZ, E., and MUCHNIK, I. An implementation of logical analysis of data. *IEEE TRANS. On Knowledge and Data Engineering*, 12 (2000) 292-306.

[20] BORWEIN, J., BRADLEY, R. Empirically determined Apéry-like formulae for zeta (4n+3). *Experimental Mathematics* 6 (1997) 181–194.

[21] BORWEIN, J.M., LISONĚK, P. Applications of integer relation algorithms. *Discrete Mathematics* 217 (2000) 65–82.

[22] BOUVIER, A., and GEORGE, M. *Dictionnaire des Mathématiques.* Presses Universitaires de France (1979). (in french)

[23] BREIMAN, L., FRIEDMAN J., STONE, C.J. and OLSHEN, R.A. *Classification and Regression Trees.* Chapman and Hall (1984).

[24] BREWSTER, T.L., DINNEEN, M.J. and FABER, V. A computational attack on the conjectures of Graffiti: New counterexamples and proofs. *Discrete Mathematics* 147 (1995) 35–55.

[25] BRIGHAM, R.C., and DUTTON, R.D. INGRID: A software tool for extremal graph theory research. *Congressum Numerantium*, 39 (1983) 337–352.

[26] BRIGHAM, R.C., and DUTTON, R.D. A compilation of relations between graph invariants. *Networks*, 15 (1985) 73–107.

[27] BRIGHAM, R.C., and DUTTON, R.D. A compilation of relations between graph invariants. Supplement 1. *Networks*, 21 (1991) 421–455.

[28] BRIGHAM, R.C., DUTTON, R.D., and GOMEZ, F. INGRID: A graph invariant manipulator. *J. Symb. Comp.*, 7 (1989) 163–177.

[29] CAGE. The chemical and abstract graph environment. Homepage: http://www.mathematik.uni-bielefeld.de/∼CaGe/ .

[30] CAMPBELL, M., HOANE, A.J. and HSU, F.M. Deep Blue. *Artificial Intelligence* 134 (2002) 57–83.

[31] CAPOROSSI, G., CVETKOVIC, D., GUTMAN, I., and HANSEN, P. Variable neighborhood search for extremal graphs 2. Finding graphs with extremal energy. *J. Chem. Inf. Comp. Sci.*, 39 (1999) 984–996.

[32] CAPOROSSI, G., DOBRYNIN, A.A., HANSEN, P., and GUTMAN, I. Trees with palindromic Hosoya polynomials. *Graph Theory Notes N.Y.*, 37 (1999) 10–16.

[33] CAPOROSSI, G., FOWLER, P.W., HANSEN, P., and SONCINI, A. Variable neighborhood for extremal graphs 7. Polyenes with maximum HOMO-LUMO gap. *Chemical Physics Letters.*

[34] CAPOROSSI, G., GUTMAN, I., and HANSEN, P. Variable neighborhood search for extremal graphs 4. Chemical trees with extremal connectivity index. *Computers and Chemistry*, 23 (1999) 469–477.

[35] CAPOROSSI, G., and HANSEN, P. Variable neighborhood for extremal graphs 5. Three ways to automate finding conjectures. *Discrete Mathematics.* (To appear).

[36] CAPOROSSI, G., and HANSEN, P. Finding Relations in Polynomial Time. In *XVIth International Joint Conference on Artificial Intelligence (IJCAI)* (Stockholm, 1999), vol. 2.

[37] CAPOROSSI, G., and HANSEN, P. Variable neighborhood search for extremal graphs 1. The system AutoGraphiX. *Discr. Math.*, 212 (2000) 29–44.

[38] CARBONNEAUX, Y., LABORDE, J.-N. and MADANI, M. Cabri-graphes: A tool for research and teaching in graph theory. In *Lecture Notes in Computer Science.* Vol. 1027, Berlin:Springer, 1995, pp. 123–127.

[39] CARO, Y. Colorability, frequency and Graffiti-119. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 27 (1998) 129–134.

[40] CHERVENKA, B. Graffiti.pc Red Burton Style – A Student's perspective. *preprint*, (2002).

[41] CHINN, P.Z. Discovery-method teaching in graph theory. *Annals of Discrete Mathematics* 55 (1993) 375–384.

[42] CHOU, S.C. Proving and Discovering Theorem in Elementary Geometrics using Wu's Method, Ph.D. Thesis, Department of Mathematics, University of Texas, Austin (1985).

[43] CHOU, S.C. *Mechanical Geometry Theorem Proving.* Mathematics and its Applications, 41, Dordrecht: Reidel, 1988.

[44] CHOU, S.C., GAO, X.S. The computer searches for Pascal conics. *Computers and Mathematics with Applications* 29 (1995) 63–71.

[45] CHOU, S.C., GAO, X.S., ZHANG, J.Z. A deductive database approach to automated geometry theorem proving and discovering. *Journal of Automated Reasoning* 25 (2000) 129–246.

[46] CHRISTIANI, N., and SHAW-TAYLOR, J. *Support Vector Machines.* Cambridge: Cambridge University Press (2001).

[47] CHUDNOVSKY, M., ROBERTSON, N., SEYMOUR, P., and THOMAS, R. Progress on perfect graphs, *Mathematical Programming* B 97 (2003) 405–422.

[48] CHUDNOVSKY, M., ROBERTSON, N., SEYMOUR, P., and THOMAS, R. The strong perfect graph tehorem, manuscript. http://www.gatech.adv/∼thomas/sqge.html .

[49] CHUNG, F. The average distance is not more than the independence number. *J. Graph Theory*, 12 (1988) 229–235.

[50] COLTON, S. Refactorable numbers – A machine invention. *Journal of Integer Sequences*, 2 (1999).

[51] COLTON, S. On the notion of interestingness in automated mathematical discovery. *International Journal of Human Computer Studies special issue on Machine Discovery*, 53 (2000).

[52] COLTON, S. Mathematics: A new domain for data mining. *IJCAI 01 Proceedings*, 2001.

[53] COWEN, R. Personal Communication at Graph Theory Day 42. DIMACS, Rutgers, November 2001.

[54] CVETKOVIĆ, D. Discussing graph theory with a computer, II: Theorems suggested by the computer. *Publ. Inst. Math. (Beograd)*, 33(47) (1983) 29–33.

[55] CVETKOVIĆ, D. Discussing graph theory with a computer, IV: Knowledge organisation and examples of theorem proving. In *Proc. Fourth Yugoslav Seminar on Graph Theory* (Novi Sad, 1983), pp. 43–68.

[56] CVETKOVIĆ, D. Discussing graph theory with a computer, VI: Theorems proved with the aid of the computer. *Cl. Sci. Math. Natur., Sci. Math.*, T. XCVII (1988), No. 16, 51–70.

[57] CVETKOVIĆ, D., DOOB, M., GUTMAN, I., and TORGASEV, A. Recent results in the theory of graph spectra. *Annals of Discrete Mathematics*, 36 (1988) 1–306.

[58] CVETKOVIĆ, D. JOVANOVIC, A., RADO SAVLIEVIĆ, Z. and SIMIĆ, S. Coplanar graphs. Univ. Beograd, Publ. Elektrotekn. Fak. Mat., 2 (1991) 67–81.

[59] CVETKOVIĆ, D., and KRAUS, L. "Graph" an expert system for the classification and extension of the knowledge in the field of graph theory, User's manual. Elektrotehn. Fak., Beograd, 1983.

[60] CVETKOVIĆ, D., KRAUS, L., and SIMIĆ, S. Discussing graph theory with a computer, I: Implementation of graph theoretic algorithms. *Univ. Beograd Publ. Elektrotehn. Fak, Ser. Mat. Fiz. No. 716 – No. 734* (1981) 100–104.

[61] CVETKOVIĆ, D., and PEVAC, I. Discussing graph theory with a computer, III: Man-machine theorem proving. *Publ. Inst. Math. (Beograd)*, 34(48) (1983) 37–47.

[62] CVETKOVIĆ, D., and PEVAC, I. Man-machine theorem proving in graph theory. *Artificial Intell.*, 35 (1988) 1–23.

[63] CVETKOVIĆ, D., and SIMIĆ, S. Graph theoretical results obtained by the support of the expert system "Graph". *Cl. Sci. Math. Natur., Sci. Math.*, T. CVII (1994), No. 19, 19–41.

[64] CVETKOVIĆ, D., and SIMIĆ, S. Graph theoretical results obtained with support of the expert system “GRAPH”– An extended survey. *(submitted)*

[65] CVETKOVIĆ, D., SIMIĆ, S., CAPOROSSI, G., and HANSEN, P. Variable neighborhood search for extremal graphs 3. On the largest eigenvalue of color-constrained trees. *Lin. and Multilin. Algebra*, 2 (2001) 143–160.

[66] DANKELMANN, P. Average distance and the independence number. *Discrete Applied Mathematics*, 51 (1994) 73–83.

[67] DE LA VINA, E. Bibliography on conjectures of Graffitti. http://cms.dt.uh.edu/faculty/delavinae/research/wowref.htm , 2000.

[68] DE LA VINA, E. Graffiti.pc. *Graph Theory Notes of New York*, XLII (2002) 26–30.

[69] DE LA VINA, E. Some history of the development of Graffiti. Submitted for publication, 2003.

[70] DI BATTISTA, G., EADES, P., TAMASSIA, R., and TOLLIS, I.G. Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry: Theory and Applications 4*, 5 (1994) 235–282.

[71] DI BATTISTA, G., EADES, P., TAMASSIA, R., and TOLLIS, I.G. *Graph Drawing: Algorithms for the Visualization of Graphs.* Prentice Hall, 1999.

[72] DOBRYNIN, A.A., ENTRINGER, R. and GUTMAN, I. Wiener index of trees: Theory and applications. *Acta Applicandae Mathematicae*, 66 (2001) 211–240.

[73] EPSTEIN, S.L. Ph.D. Thesis, Rutgers University, 1983.

[74] EPSTEIN, S.L. On the discovery of mathematical theorems. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence* (Milan, Italy, 1987), pp. 194–197.

[75] EPSTEIN, S.L. Learning and discovery: one system’s search for mathematical knowledge. *Comput. Intell.*, 4 (1988) 42–53.

[76] EPSTEIN, S.L., and SRIDHARAN, N.S. Knowledge representation for mathematical discovery: Three experiments in graph theory. *J. Applied Intelligence*, 1 (1991) 7–33.

[77] ERDÖS, P., PACH, J., and SPENCER, J. On the mean distance between points of a graph. *Congressus Numerantium*, 64 (1988) 121–124.

[78] ERDÖS, P., FAJTLOWICZ, S., and STATON, W. Degree sequences in the triangle-free graphs, *Discrete Mathematics*, 92 (1991) 85–88.

[79] FAJTLOWICZ, S. Written on the Wall. A regularly updated file accessible from http://www.math.uh.edu/∼clarson/ .

[80] FAJTLOWICZ, S. On conjectures of Graffiti – II. *Congr. Numer.*, 60 (1987) 187–197.

[81] FAJTLOWICZ, S. On conjectures of Graffiti. *Discrete Math.*, 72 (1988) 113–118.

[82] FAJTLOWICZ, S. On conjectures of Graffiti – III. *Congr. Numer.*, 66 (1988) 23–32.

[83] FAJTLOWICZ, S. On conjectures of Graffiti – IV. *Congr. Numer.*, 70 (1990) 231–240.

[84] FAJTLOWICZ, S. On conjectures of Graffiti – V. In *Seventh International Quadrennial Conference on Graph Theory.* (1995), Vol. 1, pp. 367–376.

[85] FAJTLOWICZ, S. Toward fully automated fragments of graph theory. *Graph Theory Notes of New York*, XLII (2002) 18–25.

[86] FAJTLOWICZ, S. *Fullerene Expanders, a List of Conjectures of Minuteman.* Available from the author.

[87] FAJTLOWICZ, S. On conjectures and methods of Graffiti. In *Proceedings of the 4^{th} Clemson Miniconference on Discrete Mathematics*, Clemson (1989).

[88] FAVARON, O., MAHÉO, M., and SACLÉ, J.-F. On the residue of a graph. *J. Graph Theory*, 15 (1991) 39–64.

[89] FAVARON, O., MAHÉO, M., and SACLÉ, J.-F. Some eigenvalue properties in graphs (Conjectures of Graffiti-II). *Discrete Mathematics* 111 (1993) 197–220.

[90] FIRBY, P., and HAVILAND, J., Independence and average distance in graphs. *Discrete Applied Mathematics*, 75 (1997) 27–37.

[91] FISHER, R.A. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7 (1936) 179–188.

[92] GALLAI, T. Maximum-minimum Satze uber Graphen (german). *Acta Math. Acad. Sci. Hungar.*, 9 (1958) 395–434.

[93] GANTER, B., and WILLE, R. *Formal Concept Analysis – Mathematical Foundations.* Berlin: Springer (1999).

[94] GLAS, E. The 'Popperian Programme' and Mathematics. Part 1: The Faillibilist Logic of Mathematical Discovery. *Studies in History and Philosophy of Science*, 32(1) (2001) 119–137.

[95] GLAS, E. The 'Popperian Programme' and Mathematics. Part 2: From Quasi-Empiriciam to Mathematical Research Programmes. *Studies in History and Philosophy of Science*, 32(1) (2001) 355–376.

[96] GRIGGS, J.R., and KLEITMAN, D.J. Independence and the Havel-Hakimi residue. *Discrete Mathematics*, 127 (1994) 209–212.

[97] GUTMAN, I. Total π-electron energy of benzenoid hydrocarbon. *Topics in Current Chemistry*, 162 (1992) 29–63.

[98] GUTMAN, I., and CYVIN, S. *Introduction to the Theory of Benzenoid Hydrocarbons.* Springer-Verlag, 1989.

[99] GUTMAN, I. and HOU, Y.P. Bipartite unicyclic graphs with greatest energy. *Match-Commun. Math. comp. Chem.* (43) (2001) 17–28.

[100] HÁJEK, P. and HAVRÁNEK, T. On generation of inductive hypotheses. *International Journal of Man-Machine Studies* 9 (1977) 415–438.

[101] HÁJEK, P. and HAVRÁNEK, T. *Mechanizing Hypothesis Formation. Mathematical Foundations for a General Theory*, Berlin: Springer, 1978.

[102] HÁJEK, P. and HOLEŇA, M. Formal logics of discovery and hypothesis formation by machine. *Theoretical Computer Science* 292 (2003) 345–357.

[103] HAKIMI, S.L. On realizability of a set of integers as degrees of the vertices of a linear graph. 1. *Journal of SIAM*, 10 (1962) 496–506.

[104] HANSEN, P. Computers in graph theory. *Graph Theory Notes of New York XLIII* (2002) 20–34.

[105] HANSEN, P. Degrés et nombre de stabilité d'un graphe. *Cahiers du Centre d'Etudes de Recherche Opérationnelle*, 17 (1975) 213–220.

[106] HANSEN, P., and MÉLOT, H. Variable neighborhood for extremal graphs 6. Analysing bounds for the connectivity index. *Journal of Chemical Information and Chemical Sciences*, (2002).

[107] HANSEN, P., and MÉLOT, H. Variable neighborhood search for extremal graphs. 9. Bounding the irregularity of a graph, in S. Fajtlowicz *et al.* (eds.), *Graphs and Discovery* , American Mathematical Society, forthcoming.

[108] HANSEN, P., and MLADENOVIĆ, N. Variable neighborhood search: Principles and applications. *European J. of Oper. Res.*, 130 (2001) 449–467.

[109] HANSEN, P., and ZHENG, M.L. Sharp bounds on the order, size, and stability number of graphs. *Networks*, 23 (1993) 99–102.

[110] HANSEN, P., and ZHENG, M.L. Upper bounds for the Clar number of a benzenoid hydrocarbon. *Faraday Transactions*, 88 (1992) 75–83.

[111] HANSEN, P., and ZHENG, M.L. The Clar number of a benzenoid hydrocarbon and linear programming. *Journal of Math. Chem.*, 15 (1994) 93–107.

[112] HANSEN, P., MLADENOVIC, N., and UROSEVIC, D. Variable neighborhood search for the maximum clique. *Les Cahier du GERAD*, G–2001–08, submitted.

[113] HANSEN, P., AOUCHICHE, M., CAPOROSSI, C., MÉLOT, H., and STEVANOVIĆ, D. What forms have interesting conjectures in graph theory? *Les Cahiers du GERAD*, G–2002–46, 2002, submitted.

[114] HARDY, G. *A Mathematician's Apology.* Cambridge: Cambridge University Press, 1992.

[115] HASTAD, J., JUST, B., LAGARIAS, T.C., SCHNORR, C.P. Polynomial time algorithms for finding integer relations among real numbers. *SIAM Journal on Computing* 18 (1989) 859–881.

[116] HAVEL, V., A remark on the existence of finite graphs. *Casopis Pest. Mat.* 80 (1955) 477–480.

[117] HOU, Y.P. Unicyclic graphs with minimum energy. *J. Mat. Chem.*, 29 (2001) 163–168.

[118] HSU, F.-H. *Behind Deep Blue*, Princeton: Princeton University Press, 2002.

[119] KNUTH, D. *The Stanford Graphbase: A Platform for Combinatorial Computing.* Addison-Wesley, Reading, Massachusetts, 1993.

[120] KURZWEIL, R. *The Age of Spiritual Machines.* London: Penguin, 2002.

[121] LANGLEY, P. The Computer-Aided Discovery of Scientific Knowledge. *Discovery Science: Proceedings of the First International Conference on Discovery Science. Lecture Notes in Artificial Intelligence*, 25–39, (1998).

[122] LANGLEY, P., SIMON, H.A, BRADSHAW, G.L., and ZYTKOW, J.M. *Scientific Discovery, Computational Explorations of the Creative Process.* Cambridge, Mass: MIT Press.

[123] LAKATOS, I. *Proofs and Refutations.* Cambridge, Mass: Cambridge University Press, 1976.

[124] LARSON, C. Intelligent machinery and mathematical discovery. *Graph Theory Notes of New York*, XLII (2002) 8–17.

[125] LARSON, C. On progress in the automation of mathematical conjecture-making. *preprint*, (2002).

[126] LEYDESDORFF, L. *The Challenge of Scientometrics: The Development of Measurement and Self-Organization of Scientific Communications.* Universal Publisher (2001).

[127] LI, M., AND VITANY, P. *An Introduction to Kolmogorov Complexity and its Applications.* New York: Springer, 1997.

[128] MAC LANE, S. Comment on "Theoretical Mathematics": Towards a cultural synthesis of mathematics and theoretical physics. *Bulletin of the American Mathematical Society*, 30 (1994) 13–15.

[129] MANGASARIAN, O.L. Arbitrary-norm separating plane. *Operations Research Letters*, 24 (1999) 15–23.

[130] MATHWORKS, Inc. Matlab: The Language of Technical Computing. The MathWorks, Inc.

[131] MC KAY, B.D. Nauty user's guide (version 1.5). Tech. Rep. TR-CS-90-02, Department of Computer Science, Australian National University, 1990.

[132] MCKAY, B.D. Isomorph-free exhaustive generation. *J. Algorithms*, 26 (1998) 306–324.

[133] MC CUNE, W. Solution of the Robbins problem. *J. Automated Reasoning*, 19 (1977) 263–276.

[134] MEHLHORN, K., and NÄHGER, S. LEDA: A platform for combinatorial and geometric computing. *Communications of the ACM*, 38(1) (1995) 96–102.

[135] MLADENOVIĆ, N., and HANSEN, P. Variable neighborhood search. *Computers and Operations Research*, 29 (1997) 1097–1100.

[136] OTTER. An Automated Deduction System. Web Site.

[137] PEPPER, R. On New Didactics of Mathematics-Learning Graph Theory via Graffiti. *Preprint*, (2002).

[138] POLYA, G. *Mathematics and Plausible Reasoning, Volume 1. (Induction and Analogy in Mathematics).* Princeton: Princeton University Press, 1954.

[139] POLYA, G. *Mathematics and Plausible Reasoning, Volume 2. (Patterns of Plausible Inference).* Princeton: Princeton University Press, 1954.

[140] POPPER, K. *The Logic of Scientific Discovery.* Hutchinson, London, 1959.

[141] PRICE, D. DE SOLLA, *Little Science, Big Science.* New York; Columbia University Press (1963).

[142] PUJOL, F. Étude d'un système automatisé en théorie des graphes (french). Travail de fin d'études IIE, sous la direction de Gilles Caporossi et Pierre Hansen. Rapport final. GERAD. 1999.

[143] QUINLAN, J.R. *C4.5. Programs for Machine Learning.* Morgan Kaufmann, 1993.

[144] RANDIĆ, M. On characterization of molecular branching. *Journal of the American Chemical Society*, 97 (1975) 6609–6615.

[145] RADZISZOWSKI, S.P. Small Ramsey numbers. Dynamic survey 1. *Electronic Journal of Combinatorics* (1994). Updated 1998.

[146] ROBERTSON, N., SANDERS, D., SEYMOUR, P., and THOMAS, R. The four-color theorem. *J. Combinatorial Theory, Ser. B*, 70 (1997) 2–44.

[147] ROGET'S II. The New Thesaurus@Bartleby.com

[148] ROWLINSON, P. A deletion–contraction algorithm for the characteristic polynomial of a multigraph. *Proceedings of the Royal Society of Edinburgh A*, 105 (1987) 153–160.

[149] SAATY, T., and KAINEN, P. *The Four-Color Problem: Assaults and Conquest.* New-York: Dover (1986).

[150] SAMUELSON, P.A. *Economics.* New York: Mc Graw Hill, 1968.

[151] SEYMOUR, P. Personal Communication at the Graph Coloring and Applications Workshop. CRM, Montreal, May 1998.

[152] SIBLEY, T., and WAGON, S. Rhombic Penrose tillings can be 3-colored. *American Mathematical Monthly*, (2000) 251–253.

[153] SIMIĆ, S. Some results on the largest eigenvalue of a graph. *Ars Combinatoria*, 24A (1987) 211–219.

[154] SIMIĆ, S., and KOCIĆ, V. On the largest eigenvalue of some homeomorphic graphs. *Publ. Inst. Math.* (Beograd) 40 (1986) 3–9.

[155] SHEN, W., and SIMON, H.A. Fitness Requirements for scientific theories containing recursive theoretical terms. *British Journal for the Philosophy of Science*, 44 (1993) 641-652.

[156] SKIENA, S. The Graphs of Graffiti:, *directory*, of a collection of 195 graphs from the database of Graffiti. The graphs have been converted to Combinatorica format. The database consists mostly of counterexamples, most of which were found by Noga Alon, Robert Beezer, Tony Brewster, Michael Dineen, Shui-Tain Chen, Paul Erdös, Siemion Fajtlowicz, Odile Favaron, Maryvonne Maheo, J. Riegsecker, Jean-Franois Sacle, Michael Saks, Paul Seymour, James Shearer, B.A. Smith, William Staton and Peter Winkler.

[157] SLOANE, N. The On-Line Encyclopedia of Integer Sequences, interactive Web Site.

[158] SMALE, S. Mathematical problems for the next century. *The Mathematical Intelligence* 20 (1998) 7–15.

[159] TURAN, P. An extremal problem in graph theory (in Hungarian) *Mat. Fiz. Lapok*, 48 (1941) 436–452.

[160] VAN NUFFELEN, C. A bound for the chromatic number of a graph. *American Mathematical Monthly*, 83 (1976) 265–266.

[161] WOLFRAM, Research Inc. *Mathematica Language and Software.* Wolfram Research, Inc.

[162] WOS, L. *The Automation of Reasoning: An Experimenter's Notebook with other Tutorial.* New-York, Academic Press (1996).

[163] WU, W.-T. On the decision problem and the mechanization of theorems proving in elementary geometry. *Scientia Simica* 21 (1978) 157–179.

[164] WU, W.-T. Basic principles of mechanical theorem proving in geometrics. *Journal of Systems Science and Mathematical Sciences* 4 (1984) 207–235, republished in *Journal of Automated Reasoning* 2 (1986) 221–252.