Graffiti.pc: A Variant of Graffiti

Ermelinda DeLaVina

Abstract. Graffiti.pc is a new conjecture-making program, whose design was influenced by the well-known conjecture making program, Graffiti. This paper addresses the motivation for developing the new program and a description, which includes a comparison to the program, Graffiti. The subsequent sections describe the form of conjectures and educational applications of Graffiti.pc to undergraduate research in graph theory.

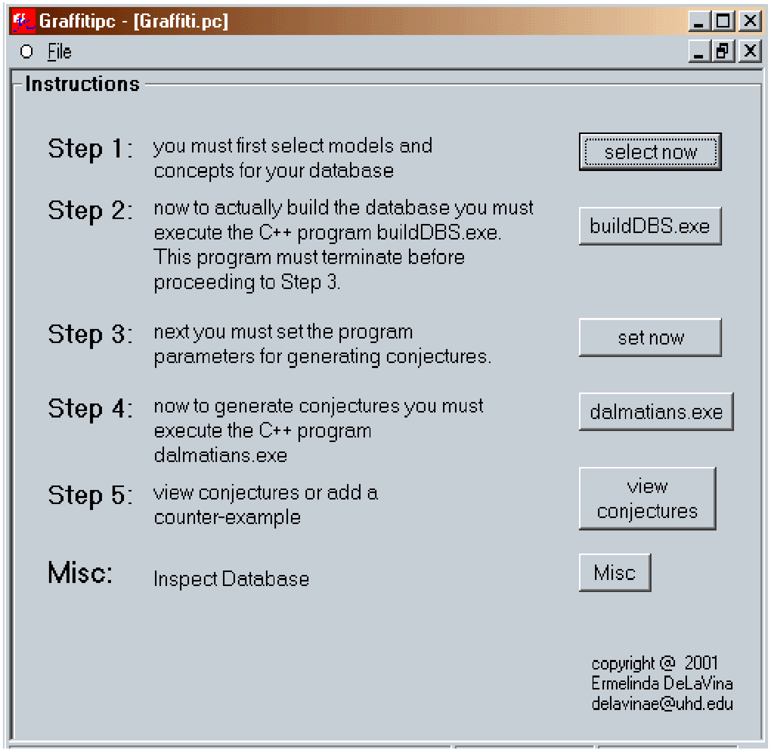
1. Introduction

Graffiti.pc is a conjecture-making computer program whose design was strongly influenced by the design of the conjecture-making program *Graffiti.* The program *Graffiti* was written in the mid-1980s by Siemion Fajtlowicz of the University of Houston. As his student in the early 1990s, the author contributed to the development of the most recent versions of *Graffiti* [**6**]. While that experience strongly influenced the design of the new program, *Graffiti.pc*, it is appropriate to note that the initial goals for the creation of the respective programs were distinct. A main short-term goal in the creation of *Graffiti.pc* was to have a user-friendly PC platform Graffiti-like program, which undergraduate students could utilize. Whereas almost from the onset of *Graffiti's* creation, Fajtlowicz was announcing conjectures to other researchers, but did not design it with other users in mind until recently [**7**]. Consequently, most comparisons of the programs, in this paper, are intended only as points of reference.

In the summer of 2001, *Graffiti.pc's* creation was realized (by the author) and this is the program under discussion in this paper. In particular, the focus will be a description of the program followed by a description of undergraduate research applications.

Key words and phrases. Graph Theory, Graffiti, Graffiti.pc, Undergraduate Research, Mathematical Discovery.

A note of gratitude is extended to the coordinators of the DIMACS Workshop on Computer-Generated Conjectures from Graph-Theoretical and Chemical Databases for the invitation to present on applications of Graffiti; with the exception of the section called recent developments, this paper is an elaboration of the topics discussed during one of the presentations. A note of gratitude is also extended to the editors of the Journal of Graph Theory Notes of New York (GTN) for their permission to reproduce the contents of the article "Graffiti.pc", which appeared in GTN in 2002 (see [**5**].) In the same volume, GTN published papers of other participants of Graph Theory Day 42, which was held the day before the DIMACS workshop.



Instructions

Step 1: you must first select models and concepts for your database

Step 2: now to actually build the database you must execute the C++ program buildDBS.exe. This program must terminate before proceeding to Step 3.

Step 3: next you must set the program parameters for generating conjectures.

Step 4: now to generate conjectures you must execute the C++ program dalmatians.exe

Step 5: view conjectures or add a counter-example

Misc: Inspect Database

copyright @ 2001
Ermelinda DeLaVina
delavinae@uhd.edu

FIGURE 1. *Graffiti.pc* Interface.

2. Graffiti.pc Description

Given a collection of mathematical objects, in this case a collection of graphs[1], which will be referred to as models, and a collection of numerical invariants, computable for each model, *Graffiti.pc* generates a system of inequalities (conjectures) between combinations of the invariants. The three integral components of *Graffiti.pc* consist of the C++ subprograms BuildDbs, dalmatians, and the Visual Basic user interface (see Figure 1.) The subprogram BuildDbs builds a two dimensional database indexed by models and invariants. The database generated by BuildDbs is input for the subprogram dalmatians, which implements Siemion Fajtlowicz's dalmatian heuristic for generating conjectures; we describe the heuristic in detail in the following section. As seen in Figure 1, the user interface provides for user selection of models and invariants, execution of the Builddbs subprogram, user selection of parameters for, and execution of, the dalmatians subprogram. In addition to the database generated by Builddbs, the subprogram dalmatians expects the user to select a xed invariant[2], and a relation (inequality or equality). Given this input,

[1]Thus far, Graffiti.pc has only been utilized with graphs as mathematical objects, however, Graffiti has been utilized with other objects such as polygons and sequences (see [**9**].)

[2]In practice, a term (algebraic express) can be xed, but this option is not available through the interface at this time.

the dalmatians subprogram generates conjectures; in section 2.2, we provide an example of this process. During or after its execution the user may view conjectures, submit counterexample(s), and choose to re-execute[3] the dalmatians subprogram.

By comparison to *Graffiti's* database support program Algernon, in 2001 *Graffiti.pc*'s BuildDbs was limited in the number of available graph theoretical invariants. At its inception, it had the capability of generating about one hundred and thirty graph theoretical invariants, most of which were on degree and distance invariants of a graph, its complement graph and its second power graph[4]. However, *Graffiti.pc*'s dalmatians subprogram very closely implements the principles of Fajtlowicz's dalmatian heuristic as described in the next section; a more detailed description is found in [**4**]. Overall, the most striking difference is *Graffiti.pc's* graphical user interface, since *Graffiti* is a Unix platform, menu and le driven program. Moreover, while both programs allow the user to select the relation, invariants and graphs for the database, another noticeable difference is that *Graffiti.pc* allows a user control over the algebraic operations utilized to generate expressions that result in conjectures.

2.1. The Dalmatian Heuristic.

Siemion Fajtlowicz's conjecture-making *dalmatian heuristic* was described by him in [**6**] as follows:

> "The program keeps track of conjectures made in the past and when it runs across a new candidate for a conjecture then rst of all it verifies if there is an example (in the database) demonstrating that the conjecture does not follow from the previous conjectures. If there is no such example then the conjecture is rejected as non-informative. If there is one, then the program proceeds with testing the correctness of the conjecture, and finally it verifies whether the conjecture should be rejected by one of its other heuristics. If the conjecture is accepted by the program then the list of conjectures is revised and those conjectures which are less informative than the new one are removed from the list and stored separately in the case the new conjecture will be refuted in the future".

As in *Graffiti*, for each conjecture selected by the program to appear on its list of reported conjectures, the number of models in the database for which the relation ($\leq$, $\geq, or =$), between the (user) selected invariant and (program) conjectured bound, is actually equality is called the *touch number* of the conjecture.

The implementation of the *dalmatian heuristic* by *Graffiti.pc* begins in the same manner as described by Fajtlowicz. The rst step in which it differs is that before accepting a relation as a conjecture, the program rst verifies if the touch number is at least the user-specified minimum touch number. The option of a minimum touch number was motivated by the experience of working on conjectures of *Graffiti*. The next step in which the dalmatian implementation differs is in the removal of conjectures. In *Graffiti.pc*, a variant of the *irin heuristic*[5] [**8**] is used first to remove conjectures if they follow by transitivity from the new conjecture. Note that in

[3]At this time, *Graffiti.pc*, unlike *Graffiti*, does not have an option for resuming the generation of conjectures.

[4]The second power of a graph $G = (V, E)$, is the graph on vertex set V such that two vertices are adjacent if and only if the vertices are at distance at most two in the graph G.

[5]*Irin* is a heuristic which rejects conjectures if they follow by transitivity from other conjectures made by the program.

Fajtlowicz's description of the dalmatian heuristic the removal of such conjectures is accomplished in any case, however for *Graffiti.pc* the decision was made to seize the opportunity to report such relations. The results are stored separately as they are not a part of the dalmatians list of conjectures. An example of such a conjecture is described in the next section.

The above-mentioned differences in the dalmatian heuristic implementations of the two programs are minor. The central ideas remain the same; put simply, the programs accept a new conjecture if it contributes something new to the current list of conjectures, and they both remove existing conjectures (from a maintained list of accepted conjectures) if the new conjecture is better. In practice, the dalmatian heuristic stops if and only if for every graph G in the database there exists a conjecture on the list whose touch number was contributed to by the graph. Thus in addition to providing a list of conjectured bounds, say c_1, c_2,..., c_k, for a user-selected term, say x, the entire list is interpreted as the following conjecture. For the sake of example, let us assume that the c_i are lower bounds on x, that is for every i, $x \geq c_i$.

For every graph G in a class of graphs (represented in the database),

$$x = \text{maximum of } \{c_1,\ c_2, ..., c_k\}.$$

2.2. Form of Conjectures. Likewise as in *Graffiti* (versions after 1992), conjectures are inequalities between terms of a Σ-Algebra, on the set of invariants in the database, together with binary, and unary operations. In *Graffiti.pc* a term is represented as a *syntax tree*, that is, a tree in which each node represents an operator and the children of the node represent the operands. At present, *Graffiti.pc* provides fourteen unary operations and ve binary operations. Examples of such operations are the reciprocal, the natural logarithm, ceiling, addition, and multiplication.

Below are three conjectures for trees (connected acyclic graphs) of highest touch number (each greater than 30% of the size of the model set) generated by *Graffiti.pc.* Conjecture 2.4 is a product of the *irin heuristic* implemented by the program as described previously. They are all correct; the rst two are easily proven and the last two are a bit more challenging as exercises. Further, since the relation in Conjecture 2.3 is valid for all simple graphs, we note that, at present, the *echo heuristic*[6] has not been implemented in *Graffiti.pc.*

For the listed conjectures, the program parameters were set as follows. The xed term was the path covering number, the relation was greater than or equal and the minimum touch was set to fty. The model set was comprised of all trees on fewer than twelve vertices, as generated by Brendan McKay's program *makeg* [**10**], and a hodgepodge of 26 other trees. The invariant set was comprised of 68 of the available invariants.

The *path covering number* of a graph G, denoted by $\rho(G)$, is the minimum number of vertex disjoint paths needed to cover the vertices of the graph. The *number of leaves* of a tree is the number of vertices of degree one. We put $\Delta(G)$ to be the maximum degree of a graph G.

CONJECTURE 2.1. If the graph G is a tree, then $\rho(G) \geq \Delta(G) - 1$.

CONJECTURE 2.2. If the graph G is a tree, then $\rho(G) \geq \left\lceil \frac{\text{number of leaves}}{2} \right\rceil$.

[6]The *echo heuristic* was described by Fajtlowicz in [**8**] as a heuristic that rejects conjectures about a property of graphs if they can be generalized to a more general property of graphs.

Let $\alpha(G)$ denote the maximum number of vertices of the graph which are pairwise non-adjacent (i.e. the *independence number* of a graph) and $n(G)$ the number of vertices of a graph.

CONJECTURE 2.3. If the graph G is a tree, then $\rho(G) \geq 2\alpha(G) - n(G)$.

Let $E(v)$ be the number of vertices at even distance from vertex v. Let E_{max} be the maximum of $E(v)$ over all vertices of the graph, and similarly let E_{min} be the minimum of $E(v)$ over all vertices of the graph.

CONJECTURE 2.4. If the graph G is a tree, then $2\alpha(G) - n(G) \geq E_{max} - E_{min}$.

The main objective for listing the conjectures in this paper was to demonstrate the algebraic form of conjectures. We observe that it seems that the main difference between an educational and research version of the program is the simplicity of the invariant set. Thus, given *Graffiti.pc*'s limited invariant set, only conjectures for which the dalmatians program reported a significantly high touch number were reported. From experience, as was the case in this execution of the program, it seems that conjectures of high touch number (relative to the model set) are usually correct.

On a technical note, the first two conjectures appeared almost immediately and a while later the program reported that $\rho(G) \geq E_{max} - E_{min}$, which several hours later was replaced by the third conjecture listed above. The program did not stop by reaching the halting condition. The program execution was interrupted (after it ran for a day), at which time there were 120 trees in the database that did not contribute to the touch number of any conjecture on the list.

3. Educational Application

Graffiti.pc's initial application was primarily educational. In particular, undergraduate students have used the program's conjectures as the topic of their senior projects[7]. Barbara Chervenka's project was completed in December 2001; her activities and results are described in this paper. The rst phase of her project was to resolve conjectures, which were lower bounds on the sum of the independence number and the clique number[8] of a graph. Courtesy of Siemion Fajtlowicz and the University of Houston Mathematics Department, conjectures of this phase of her project were generated by *Graffiti* on that campus' alpha computers.

At about the time that the topic for conjectures was changed, which was also about the same time that *Graffiti.pc* was created, Fajtlowicz announced a set of rules to follow while working on conjectures. The rules are called the *Red Burton rules* [**7**]. With the previously mentioned changes in place, Chervenka began the second phase of her senior project. The input for the program was the database, which was composed of the complete graph on one vertex as the model set, and the *alpha-core number* (the number of vertices common to all maximum independent sets) and invariants of the degree sequence of a graph as the invariant set. The xed invariant was the alpha-core number, and the relation was greater than or

[7]Senior projects at the University of Houston-Downtown are intensive studies under the guidance of a member of the mathematics faculty which culminate in an individually researched and formally written report and oral presentation.

[8]The *clique number* of a graph is the maximum number of vertices of the graph which are pairwise adjacent.

equal. A modification of the Red Burton rules, which are described below, were utilized for this project.

1. The rst conjecture to appear on the list will be resolved. (Note that in *Graffiti.pc*, if the conjectures remain unsorted by touch number then it is usually the case that the first is the most simply stated conjecture).
2. If the resolved conjecture is false then nd the minimum number of vertices in a counterexample, and next the minimum number of edges of a counterexample with the minimum number of vertices. In this case, the counterexample is added to the database.
3. If the resolved conjecture is true then characterize the case of equality and determine if one can verify in polynomial time that a graph has the characterization described. In the case that such a characterization is accomplished, graphs from the class are forbidden from the database; further, any counterexamples for subsequent conjectures cannot be in this class of graphs. Otherwise, the next conjecture on the list is resolved.

After about two months into the second phase of the project, her partial result was a characterization of the alpha-core number in terms of concepts involving only the degree sequence of a graph for the class of graphs comprised of stars, complete graphs, complete graphs minus an edge, complete graphs minus a triangle, and complete graphs minus a triangle and minus an edge disjoint from the triangle. Her result is stated as follows.

Let G be a simple graph, $E(G)$ its set of edges, and $\overline{G}$ the complement graph of G. Let K_m denote a complete graph on m vertices, and let $\overline{K_m}$ denote the m-vertex graph with no edges. Let $K_{3,2}$ denote the complete bipartite graph with the parts having 3 and 2 vertices, respectively. The *join* of graphs G and H is the graph obtained from the union of G and H by adding edges $\{u, v\}$ where u is a vertex of G and v is a vertex of H. The *alpha-core number* of G, denoted by $\alpha_c(G)$, is the cardinality of the intersection of all maximum independent sets of the graph G. The *length of a graph G* is de ned as the square root of the sum of the squares of degrees of the vertices of G.

Theorem 3.1 (Chervenka [**2**]). *If G is a simple connected graph,*

$$\alpha_c(G) = \begin{cases} E(G) & \text{if } G \simeq join(K_1, D_m) \text{ for } m \geq 2 \\ 2\ E(\overline{G}) & \text{if } G \simeq\ K_m \text{ or } join(K_m, D_2) \text{ for } m \geq 2 \\ \lfloor Length(\overline{G}) \rfloor & \text{if } G \simeq join(K_m, D_3) \text{ for } m \geq 2 \text{ or} \\ & join(K_m, K_{3,2}) \text{ for } m \geq 0 \\ ? & \end{cases}$$

At the end of her project the pending list (List 11) of conjectures for simple connected graphs was as follows:

If G is not isomorphic to K_m, join(K_m, D_2) for $m \geq 2$, join(K_1, D_m) for $m \geq 2$, join(K_m, D_3) for $m \geq 2$ nor join$(K_m, K_{3,2})$ for $m \geq 0$, then

1. $\alpha_c(G) \leq 1+$ 2nd smallest element in the set of degrees of G.
2. $\alpha_c(G) \leq 1+$ the maximum degree of the complement of G.
3. $\alpha_c(G) \leq 2 *$ (the frequency of the maximum degree of G).

Formally, the project ended; nevertheless, Chervenka determined that the first and last conjectures are false and the second is true. The next step called for the determination of a smallest counterexample to the rst statement.

During the rst phase of the project, the goal of working on conjectures was simply to resolve as many as possible. However, through some appropriate prompting of the student, the program's response to counterexamples was emphasized. As a result, she eventually learned to look for families of counterexamples, special classes of graphs on which the conjecture may be true, and to characterize the case of equality for a proven conjecture. Moreover, in the early phase of the project she was encouraged to nd counterexamples on the smallest possible number of vertices and then the smallest number of edges on that number of vertices. Early on, one motivation for this was to provide statements relevant to conjectures, that required proof, but eventually in the second phase of the project, this skill was compulsory. By the end of the project, Chervenka had examined, in varying degrees, over 80 conjectures, and almost half were resolved. The chronology and details of which are described in her senior project report [**2**]. Aside from the obvious difference of the selected term, the first phase and second phase of her project differed in that the conjectures generated by *Graffiti* were announced by the author (as the research advisor), whereas by list 3 of the second phase, she was reporting the conjectures as she was the user of *Graffiti.pc*.

In addition to providing the previously cited student research opportunities, another advantage of using *Graffiti.pc* (and *Graffiti*) as a pedagogical tool was that, by the nature of how the programs were utilized, the difficulty level of conjectures increased as the students' knowledge and the number of graphs in the database increased. Further, as an educator, the abundance of good problems accessible to students, even undergraduate students, was stimulating. But the potential is even greater as Fajtlowicz observed in [**7**], "If the students wish to, they may, run the program according to their own rules or simply by working on conjectures of their own choice, ending up with highly personalized exercises and problems".

4. Recent Developments

Most changes to the 2002 "Graffiti.pc" paper [**5**] were minor; however, some comments and explanations were added for clarity in the Dalmatian Heuristic and Form of Conjectures sections. There have been recent developments in the implementation of *Graffiti.pc* as an educational tool. Firstly, since the publication of [**5**] three other students[9] have completed their senior projects on graph theory utilizing *Graffiti.pc*; included in this section are descriptions of two results of those projects. Secondly, as a consequence of our participation in the DIMACS workshop, Gunnar Brinkman utilized *Graffiti.pc* in a graduate graph theory course (for pre-service teachers) during the spring of 2002; and in the spring of 2004, he conducted a workshop utilizing *Graffiti.pc* at the University of Bielefeld in Germany for teachers of advanced high school students.

4.1. On the Number of Triangles of a Graph. One senior project included an investigation of the graph invariant the *number of triangles* (i.e. the number of K_3's) of a graph. The motivation for adding this graph invariant to *Graffiti.pc* came after a special case of a problem (described in [**3**]) of Paul Erdős (and generalized

[9]The students were Kelly Wroblewski who graduated Dec. 2002, Laura Salazar and Zahra Salehpoor who both graduated May 2003

by Béla Bollobás) was discussed during the DIMACS workshop. The statement of the problem is as follows, "what is the maximum number of K_4 subgraphs a simple graph can contain if it has at most x triangles?". This project began by investigating upper bounds on the number of K_3's of a graph. Since as usual, the investigation began with only K_1 in the database of graphs, many conjectures were disproved (some by nding a smallest counterexample), families of counterexamples were determined, and some conjectures were proven (some only for special cases). The highlight result of this project, was a proof of a special case of the Erdős-Hanani result, see [**3**]. The statement proven in the project is that the maximum number of K_3 subgraphs a simple graph can contain is 5, if it has at most 8 edges. In the notation of Bollobás, the result is $k_3(k_2 \leq 8) = 5$. While this, of course, is a special case of a known result (but certainly very appropriate for undergraduate research), the reason that it is of special interest here is that one of *Graffiti.pc's* conjectures was used as a lemma for the proof. This proven conjecture of *Graffiti.pc* is given as the following simple proposition, after the necessary notation is provided. Let G be a simple graph, and v a vertex of the graph. Let $T(v)$ be the number of triangles incident to the vertex v and T_{max} the maximum of $T(v)$ over all vertices of G. Lastly, let $T(G)$ denote the number of triangles of the graph G.

Proposition 4.1. For G a simple graph, $T(G) \leq \frac{n(G)*T_{max}}{3}$.

We note that this conjecture of *Graffiti.pc* did not make it to the final list of conjectures on that execution of the program; due to a numerical error, it was being reported as one of the best conjectures for a period in the program. In any case, the student promptly realized that it was correct, and used it to prove that $k_3(k_2 \leq 8) = 5$ [**11**]. The conjecture that eventually replaced it (after numerical errors were dealt with) was the following stronger statement, which we observe is a special case of a generalization of the Handshaking Theorem[10].

Proposition 4.2. For G a simple graph, $T(G) = [\sum_{v \in V(G)} T(v)]/3$.

4.2. Related to Graffiti's #158. During one instance of another senior project, a student was resolving conjectures on the *independence number* of a graph. Of particular interest was the following conjecture encountered in one of *Graffiti.pc's* lists, which the student subsequently proved; it is listed next as a proposition. For G a simple graph, we let $\alpha(G)$ be the independence number of the graph G and $\Delta(G)$ the maximum degree of G.

Proposition 4.3. Let G be a simple graph. If G is not an empty graph, then

$$\alpha(G) \leq \text{the maximum of} \{\Delta(G), \Delta(\overline{G})\}.$$

Let G be a simple graph, and let $\delta(G)$ be the minimum degree of G. In *Written on the Wall* [**9**] conjecture number 158, which is known to be correct[11], is presented here as the following proposition.

[10]In most discrete mathematics textbooks one can nd the statement that twice the number of edges of a graph is equal to the sum of the degrees; a obvious generalization is that for k an integer such that $k \geq 2$,, k times the number of complete k-vertex subgraphs of a graph is equal to the sum of the number of k-vertex subgraphs incident to vertices.

[11]It was listed between two sets of ***, which indicates that it was considered an exercise.

PROPOSITION 4.4. Let G be a simple graph. Then

$$\alpha(G) \leq n(G) - \delta(G).$$

This inequality is easily shown to be equivalent to $\alpha(G) \leq 1 + \Delta(\overline{G})$, which the student had also proven, as it was on a list of *Graffiti.pc's* conjectures. Moreover, she used Proposition 4.4 (in the latter equivalent form) to prove Proposition 4.3. We observe that Proposition 4.3 (*Graffiti.pc's* conjecture) follows from the case of equality (proven by Brewster et. al. in [**1**]) of Proposition 4.4. However, what seemed interesting (to the author) about $\alpha(G) \leq$ the maximum of $\{\Delta(G), \Delta(\overline{G})\}$ is that its proof (as opposed to the research advisor) prompted the student to consider the case of equality of $\alpha(G) \leq 1 + \Delta(\overline{G})$.

References

[1] T. Brewster, M. Dineen and V. Faber, Computational attack on conjectures of Graffiti: New counterexamples and proofs, *Discrete Mathematics*, **147**, (1995), 1-3.

[2] B. Chervenka; Graph theory Graffiti/Graffiti.pc style, Senior Project Report, University of Houston-Downtown, Houston, Texas 77002 (2001).

[3] R. Cowen and W. Emerson, On finding $k_4(k_3 \leq x)$, *Graph Theory Notes of New York*, **XXXIV:7**, New York Academy of Sciences, (1998), 26-30.

[4] E. DeLaVina, On some history of the development of Graffiti, (2004), preprint.

[5] E. DeLaVina, Graffiti.pc, *Graph Theory Notes of New York*, **XLII:3**, New York Academy of Sciences, (2002), 26-30.

[6] S. Fajtlowicz, On conjectures of Graffiti V, *Proceedings of the Seventh Quadrennial International Conference on the Theory and Applications of Graphs*, **1** (1995), 367-376.

[7] S. Fajtlowicz, Toward fully automated fragments of graph theory, *Graph Theory Notes of New York*, **XLII:2**, New York Academy of Sciences, (2002), 18–25.

[8] S. Fajtlowicz, On conjectures of Graffiti III, *Congressus Numerantium*, **66** (1988), 23–32.

[9] S. Fajtlowicz, Written on the Wall, A list of conjectures of *Graffiti*, accessible at http://www.math.uh.edu/siemion.

[10] B. McKay; *makeg*, a computer program, accessible at
http://www.theory.csc.uvic.ca/ cos/gen/grap.html.

[11] K. Wroblewski, A few topics in graph theory, Senior Project Report, University of Houston-Downtown, Houston, Texas 77002 (2002).

DEPARTMENT OF COMPUTER AND MATHEMATICAL SCIENCES, UNIVERSITY OF HOUSTON - DOWNTOWN, HOUSTON, TEXAS 77002

E-mail address: `delavinae@uhd.edu`