SOME HISTORY OF THE DEVELOPMENT OF GRAFFITI

ERMELINDA DELAVINA

ABSTRACT. This paper provides some history of the development of the conjecture-making computer program, Graffiti. In the process, its old and new heuristics are discussed and demonstrated.

1. INTRODUCTION AND MOTIVATION

Graffiti is a conjecture-making computer program written in the mid-1980s by Siemion Fajtlowicz. Since its inception, Graffiti’s conjectures have inspired about eighty papers, some by researchers such as Alon, Babolás, Chung, Erdős, Kleitman, Lovász, Pach, Seymour, Shearer and Spencer, and parts of five Ph.D. theses (see [10]). For Fajtlowicz, the main interest in the automation of some of the conjecture-making processes was to understand what makes a good conjecture. This idea is a recurrent theme in the series of papers [24, 25, 26, 27, 28, 29, 30], in which heuristics (methods for deciding which relations are interesting) are described and some conjectures are announced.

A motivation for the paper at this time is that interest in technical details not mentioned in Fajtlowicz’s papers has increased as the use of computers in mathematical discovery expands. In particular, as computerized mathematical discovery develops into a discipline, papers on the processes of mathematical discovery programs provide useful resources to researchers with similar interests; see [50] for some history on mathematical discovery programs. Since announced open conjectures seem relevant to a program being identified as a conjecture-making program, we include discussion of the two main conjecture-making versions of Graffiti referred to in this paper as the 1980s version and the 1990s version (see Figure 1 for an overview). Most of Graffiti’s announced conjectures are listed in the document Written on the Wall [31] or in various papers such as [11], [12], and [32]. Conjectures numbered 1 through 746 in [31] were made by the 1980s version. All but a couple of the others were generated by the 1990s version, which include those listed in Conjectures of Minuteman [32] (conjectures about fullerenes) and Pony Express [33] (conjectures about benzenoids).

As a student of Fajtlowicz in the early 1990s, the author’s main contributions were to the development of the Dalmatian version of Graffiti, first described in [30]. The resources utilized to provide a description of the 1980s version of Graffiti and some history of its development were mainly Fajtlowicz’s papers [25, 26, 27, 28, 29], the program code, experimentation, and discussions with Fajtlowicz. The 1980s version of the program is described first; demonstrations of heuristics and discussion of ancillary features are included. Next, the development of two intermediate versions are described as they lead to the design of the Dalmatian version; in this paper, all

Key words and phrases. Graffiti, Graffiti.pc, Graph Theory, Mathematical Discovery.
of these are collectively referred to as the 1990s version of the program. We then describe the Dalmatian heuristic of Graffiti, and provide a demonstration of the Dalmatian heuristic implemented by Graffiti.pc.

2. THE 1980s VERSION

In the early 1980s, Fajtlowicz proposed to a graduate student the task of writing a computer program that would make mathematical conjectures. The student was Shui-Tain Chen, and the program was called Little Paul. Although, Chen did not collaborate with Fajtlowicz on the development of the (distinct) program that a couple of years later would be known as Graffiti, in 1990 she completed her Ph.D. thesis titled On Selected Conjectures of Graffiti [5].

In Fajtlowicz’s On Conjectures of Graffiti [25], submitted in mid-1986, which appeared in 1988 in Discrete Mathematics he described Graffiti as follows:

The basic idea of Graffiti is that it “knows” certain graphs and it is capable of evaluating certain formulas from graph-theoretical invariants. If none of the graphs with which Graffiti is familiar is a counterexample to a formula then the formula is considered to be a conjecture. At present Graffiti is capable of computing about 60 invariants and it performs several functions but I shall describe here only those which are relevant to conjectures described in the next section. The conjectures are of the form $I \leq J$, $I \leq J + K$ and $I + J \leq K + L$, where literals run over 20 distinct selected invariants plus a constant invariant 1. ... more than half of the program consists of various heuristics whose purpose is deletion of trivial and otherwise noninteresting but true conjectures.

This general description captured the state of development of the program until the early 1990s. Before providing further details of the general scheme of the 1980s version and a summary of its heuristics, we discuss the computer hardware resources available over time and provide a short description of the structure of Graffiti’s code.
2.1. **On the Code of Graffiti.** *Graffiti* is a Pascal, module-based, menu-driven program, whose creation was realized on a University of Houston multi-user DEC-station (whose CPU speed was by today’s standards very slow): the disk space available to a user of the computer was probably around 10 KB. In 1990, the program was moved to a faster (but of course still slow by today’s standards) VMS/VAX station\(^1\), named Charly, with 8 MB of RAM. Still, at the time the major advantage was that the machine was dedicated to executing *Graffiti*. The program remained on Charly, until 1995, at which time the code was ported from VMS to Unix (a University of Houston mathematics department multi-user machine). The development of the program over the years was, in part, driven by the resources available.

The conjecture-making code of *Graffiti* is composed of almost two-dozen modules. Each module comprises procedures implementing a given functionality; for example, one module was dedicated to procedures for input, output and updating of the database (of models and invariants), others for procedures implementing heuristics and yet others for manipulating particular user-defined types.

For a short time after *Graffiti’s* creation, all procedures for composing graphs, computing invariants and generating conjectures were part of one program. Soon afterwards, due to memory limitations, the code could no longer be compiled in one section and was split into two separate programs. The conjecture-generating code remained in the collection of modules called *Graffiti*; and the graph generating and invariant computing code is contained in a collection of modules called *Algernon*, which is discussed in the next section. In his series of papers on *Graffiti*, Fajtlowicz describes the system of two programs as *Graffiti*, with one exception, namely in

---

\(^1\)This was part of the Advanced Research Project grant, 0033652085-ARP.
ERMELINDA DELAVINA

[28] where Algernon is called a sub-program of Graffiti. At other times, he and the author refer to the conjecture-making code as Graffiti. In this paper, from this point on, the term Graffiti-System\(^2\) will be used to refer to both programs.

2.2. On the Code of Algernon. As seen in Figure 2, the input for Graffiti, called a database, can be visualized as a 2-dimensional array, indexed by models and invariants. The task of constructing a database is performed by the sub-program Algernon. Similar to Graffiti’s code, Algernon’s code is module-based, and each module comprises procedures implementing a given functionality. Two modules, called library and data of Algernon, are noted as there are multiple interchangeable versions of each, and will be referenced again in Section 4.1. Most versions of the library module are composed of a (usually) long list of short segments of code (usually one segment corresponds to a graph) with boolean switches for determining inclusion of the graph into the database. Since the library module is readily interchangeable, many such modules were developed over the years. A similarly formatted module, called data, determines which invariants are used in constructing the database.

In the 1980s, most graphs generated by Algernon were built using Pascal functions.\(^3\) For example, a star on \(n\) vertices can be generated as the join\(^4\) of an empty graph on 1 vertex and an empty graph on \(n - 1\) vertices. Algernon contains many standard graph operations, as well as some non-standard graph operations. We note that the current version of Algernon provides for other methods of including graphs in the database, such as reading adjacency sequences for graphs [35], reading adjacency lists for graphs from Gunnar Brinkmann’s CaGe program [4], Brendan McKay’s makeg program [53], and Steven Skiena’s database [55] of some counterexamples to conjectures of Graffiti. All functions and procedures for evaluating invariants are contained in Algernon, with one exception; for the duration of the 1980s, the Fortran library EISPACK (for computing eigenvalues and eigenvectors) was linked to Algernon with the assistance of Edward Dean of the University of Houston. We note, that the current version (since the late nineties) no longer utilizes EISPACK, instead the program utilizes eigenvalue procedures coded by Fajtlowicz.

2.3. On 1980s Conjecture-Making Scheme and Heuristics. For the 1980s version of the program, as Fajtlowicz described in [25] and [26], a list of certain types of formulas (most of which the user does not see) is generated, immediately after which heuristic(s) are applied. In this version, the purpose of a heuristic is deletion of trivial and otherwise noninteresting formulas, the correctness of which has been verified with respect to the database of graphs and invariants. Further, he described that the types of formulas generated by this version are of the form \(I \leq J, I \leq J + K\) and \(I + J \leq K + L\), where literals run over invariants available in the database and a constant of one. Before proceeding with this discussion, we provide a graph theoretical definition.

\(^2\)In the past, this informal naming convention seems to have contributed to confusion regarding which stage of the process handles the computation of invariants (see Figure 2).

\(^3\)In Section 5.1 we note another method.

\(^4\)The join of two 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 \(H\).
**Definition 2.1.** The *girth* of a graph with a cycle is the length of its shortest cycle. A graph with no cycle has infinite girth (in the case of *Graffiti* it is taken to be undefined). A graph is called *triangle-free* if its girth is not equal to three.

In *Graffiti* (and throughout this paper), the terms *property* and *background* are used to indicate classes of graphs satisfying specific relations between invariants. For example, suppose that one were interested in setting the program to generate conjectures about “triangle-free” graphs; in *Graffiti*, the *property* would be the collection of graphs (in the database) satisfying the relation that the girth of the graph is not equal to three; and the *background*, if one wished to use “simple graphs”, would be the collection of all graphs (in the database), which satisfy any identity. These collections of graphs are used to generate conditional conjectures of the form “if *G* is in the *property*, then some relation (on invariants) holds”. A discussion of what role a *background* may have in such a conjecture is presented for the third heuristic described below.

Figure 2 captures the general conjecture-making process of the 1980s version, which we summarize as follows. Given as input a database of graphs and their computed invariants, a user-selected (conjecture-generating) command, a *property* and (possibly) a *background*, *Graffiti* generates a list of certain types of formulas, immediately applies the heuristic(s), and lastly reports conjectures to a file or to the screen.

In the first three of Fajtlowicz’s series of papers, *On Conjectures of Graffiti* [25, 26, 27], four heuristics of this version of *Graffiti* are discussed. We provide a summary of the four heuristics, *irin*, *cncl*, *echo* and *beagle*, described in print, and of those heuristics visible in the help menu of *Graffiti* (Figure 3). The program’s code suggests that updating the help menu ceased early on. We note that there are many conjecture-making commands (in the code and thus available to the program), which are combinations and variations of the other heuristics.

- The command *ineq* as described in the help menu (Figure 3), reports all relations between invariants available in the database (of the form *I* ≤ *J*) satisfied by the models in the database. This command is frequently employed by other heuristics.
In the paper *On Conjectures of Graffiti* [25], Fajtlowicz described *irin*, which deletes those conjectures which by transitivity follow from others; and *cncl*, which deletes those conjectures of the form $I \leq J + K$ and $I + J \leq K + L$ in which one of the invariants on the left is always smaller than an invariant on the right. In practice, *irin* calls *ineq* and rejects those inequalities that follow by transitivity.

In the paper *On Conjectures of Graffiti II* [26], he described the heuristic *echo*, which implements the idea that a conjecture about a class of objects $A$ (a *property*) is considered noninteresting if it can be generalized to a larger class $B$ (a *background*). In practice, the user is prompted for a *property* and *background*. A relation is accepted (as a conjecture) if it is correct with respect to the models in the *property* but not for the models of the *background*. For example, an inequality between invariants is deemed noninteresting for triangle-free graphs if it is also true for all simple graphs.

In the paper *On Conjectures of Graffiti III* [27], a discussion of the heuristic *beagle* was given. It implements the idea that conjectures involving concepts of a different type are more likely to be interesting. While this heuristic is not explicitly listed in the help menu, it is launched by other heuristics as shall be demonstrated in Section 3. In the 1980s, the convention implemented for invariant names served some of the heuristics used by *Graffiti* (for example, those that launch *beagle*). In [27], Fajtlowicz described the representation of an invariant name by a rooted tree, which tracks the “genealogy” of the invariant (see Figure 4). Further, he described experimenting with several distance functions, and thus we note that the one described next may or may not have been the one he decided on (if there was one); in any case, we describe one that was found in the code of *Graffiti*. Using a genealogy tree representation, the *distance between two invariants* is their graph distance in the genealogy tree, if they share a common root; otherwise, the distance is defined to be the sum of the depths in their corresponding genealogy trees. For example, the invariants “*minimum degree of the graph*” and “*maximum degree of the graph*” are at distance 2 (see Figure 4), the invariants “*maximum degree of the graph*” and “*maximum of local independence of the graph*” are at distance 4, whereas “*maximum degree of the graph*” and “*girth of the complement graph*” are at distance 5.

The first heuristic on the help menu (Figure 3), but not discussed in [25], [26] or [27] is *eiff*. The heuristic *eiff* implements the idea that the fewer models in the *background* for which the relation holds, the more likely the relation is to be of interest for the models in the *property*. In practice, once this command is selected, the user is prompted to define a *property* and a *background*, a
relation and a probability. For \textit{eiff}, the probability associated with a relation is the ratio of the number of models in the \textit{background} for which the relation holds to the number of models in the \textit{background}. For each relation (candidate conjecture), if the probability associated with the relation is not more than the probability entered by the user, the relation is accepted as a conjecture.

- The heuristic \textit{cheq}, mentioned in the help menu (Figure 3), implements the \textit{echo} heuristic for the equality relation. In practice, once this command is selected, the user is prompted to define a \textit{property} and a \textit{background}, and a lower bound for the distance (as defined above) between invariants on the left and right hand side of the relation.
- The heuristic \textit{erin} as described in the help menu (Figure 3), implements the \textit{echo} heuristic for (the relation \textit{\textasciitilde}) and then applies the heuristic \textit{irin}.
- The heuristic \textit{erie} as described in the help menu (Figure 3), implements the \textit{irin} heuristic and then the heuristic \textit{echo}.
- The heuristic \textit{equus} as described in the help menu (Figure 3), reports all equalities between invariants satisfied by the models in the database.

3. Demonstrations of the 1980s Version

While the 1980s version is still executable (and thus a demonstration was possible), we note that many of that version’s heuristics are no longer used. The reader will be alerted to one instance (in Section 3.6) in which that version of the program does not currently work as it did in the past due to changes of direction in the development of the program. Further, we note that the code of the heuristics was not altered for the demonstrations in this paper; in one case code was introduced as noted in Section 3.9, and in some cases, diagnostics were included (but removed before capturing screen images) to insure the author’s understanding of some details.

Once a database is available for input and execution of \textit{Graffiti} is initiated, the first prompt is for a command. (See Figure 2 for the structure of the program.) If the user types “help”, the screen in Figure 3 appears. As previously described, some of the command options, such as \textit{echo}, \textit{irin}, \textit{cheq}, \textit{erin}, \textit{irie}, and \textit{rest}, listed in the help screen of Figure 3 are heuristics (and combinations of them) of the 1980s version as described in [25] through [27]. Other command options that are visible on this menu, but not discussed in Fajtlowicz’s papers, are for viewing and modifying the database, and for viewing conjectures (modifying them is not an option). These ancillary commands include \textit{ftch}, \textit{lisc}, \textit{lisg}, and \textit{modi}.

The following subsections include a discussion of the form of databases of graphs and invariants used for the demonstrations, a discussion on the form of \textit{Graffiti}’s output and Fajtlowicz’s method for announcement of conjectures. The demonstrations include the ancillary commands called \textit{ftch} and \textit{modi}; and the conjecture-generating commands \textit{irin}, \textit{icnl}, \textit{dirn} and \textit{erie}. The conjecture-generating commands that are demonstrated launch the heuristics, \textit{irin}, \textit{cncl}, \textit{echo}, \textit{beagle} and combinations of them. As far as the author is aware, the demonstrations in this paper are not duplications of previous executions of the program. However, we note that conjectures previously made by \textit{Graffiti} and announced in [31] do appear in demonstrations, since some similar invariants are used in the demonstrations; the reader will be alerted to such conjectures, and to other conjectures as they
demonstrate the heuristic or as they seem of interest.

3.1. Database for 1980s Version Demonstration. For the 1980s version, the database of graphs and their computed invariants was limited (due to computer memory issues) to about 200 graphs and 200 invariants. These numbers were increased and decreased at different times; however, due to the other large data types of this version (for example those used to hold candidate conjectures) they were usually limited to about 200. For the initial demonstration in this paper, the database was composed of 19 invariants and 164 simple connected graphs. On the subject of databases used to generate conjectures, Fajtlowicz reports that he rarely used more than 80 graphs in any one session of this version of Graffiti. While experience has shown that it is not essential to use a large number of graphs in order for the program to generate conjectures that are correct and interesting to researchers (see [31]), the first 80 graphs (of the 164 used for the database in this demonstration) were selected since they were readily available, and the remaining 80 graphs were taken from Skiena’s database (of some collected counterexamples to some of Graffiti’s conjectures). Invariants chosen for the demonstration included at least four computationally challenging invariants (independence number, local independence number, path covering number, and bipartite number) and many invariants on degrees and distances of graphs; definitions are provided in Section 3.4.5

We note that courtesy of the mathematics department at the University of Houston - Central Campus, the following demonstrations of the 1980s version of Graffiti were performed on one of its Unix machines. This was not the case for the demonstration of Algernon. The databases of graphs and their computed invariants (for the demonstrations) were generated by the similar program Graffiti.pc (see [13] and Section 5.2) and formatted for input to Graffiti. Other details, as they relate to some heuristics, on the form of the database are provided in later sections. But first, we briefly discuss the program’s output and the convention used by Fajtlowicz to announce conjectures.

3.2. Graffiti’s Output and Announcing Conjectures. The first public announcement of conjectures was in 1986 at the Southeastern Combinatorics and Graph Theory Conference. Eventually the 18 conjectures, announced at the conference and subsequently announced in his paper On Conjectures of Graffiti [25], became conjectures 1 through 18 in the document now known as Written on the Wall [31]. As will be seen in the following sections, Graffiti’s output, the list of conjectures (see Figure 8 for a sample), is generally directed to a file. A selection process for which conjectures of the 1980s version were added to [31] was first described in [25]. The process that Fajtlowicz described in [25] is that some of the conjectures were evaluated, counterexample(s) to some conjectures were reported to the program, the program was re-executed and again some of the conjectures were evaluated. It follows from the description in [25] that after a few rounds of this process, he announced some of the conjectures made by Graffiti. This iteration process will be demonstrated in Sections 3.7 and 3.8.

Regarding his selection of which of the conjectures of the 1980s version of Graffiti to announce, in the paper On Conjectures of Graffiti II [26], Fajtlowicz explained

5These 19 invariants are a subset of the invariants that will be used in later demonstrations (beginning with the second demonstration of Section 3.7).
Figure 5. Graffiti’s select property menu.

that “The two main reasons why I do not announce more conjectures of Graffiti is
that I can’t estimate their value and I do not want to communicate too many true
but trivial conjectures”; and later in On Conjectures of Graffiti III [27] and in On
Conjectures of Graffiti V [30], he re-iterated this while discussing the 1980s version
of Graffiti.

3.3. Demonstration of \textit{ftch}. The ancillary command, \textit{ftch}, is used to view the
database of models and their computed invariants. The menu in Figure 5 appears
once \textit{ftch} is invoked. At this point, a user can select to view a subset of the input.
Each choice available in the menu (in Figure 5), is designed to facilitate selection
of models with user-specified properties (defined by relations between invariants).
In practice, each of the options\footnote{Although, 13 and 14 were clearly added later, they are included here since they are obviously options similar to the first 9. The two choices 11 and 12 for selecting a \textit{property} were added after 1990, and thus will be discussed in Section 4.4.} 1-9 and 13-14 use the following convention: $i_1$ and
$i_2$ denote model invariants by their numerical identifiers and $c$ denotes a constant.

For our demonstration, suppose that a user seeks to view information about
“triangle-free graphs” of the database. Before using \textit{ftch}, one types the command
\texttt{listc} to list all invariants present in the database along with their corresponding
numerical identifiers, and (one) notes the value corresponding to girth, (17 in the
example database). Next, one enters the command \textit{ftch}, selects option 5 (see menu
in Figure 5) with 17 as the value for $i_1$, the number 3 as the constant (for our
example), and then enters 0 in order to select the complement of the set of graphs
for which the girth is equal to three. The selected graphs are those with girth not
equal to three (i.e. triangle-free graphs). After the selection of one or more relations
used to define a \textit{property}, the \textit{ftch} menu appears (Figure 6). Option 1 allows for
viewing the values of all invariants of a particular graph (model); option 2 allows
for viewing the values of an invariant for all of the graphs (models) selected by a
user in the previous step. The menu shown in Figure 6 lists these options and a
variety of others for inspecting the database.

3.4. A Demonstration of \textit{irin}. The first demonstration of a conjecture-making
heuristic is a demonstration of \textit{irin} with the \textit{property} selected as “simple connected
graphs” (as seen in Figure 7). As Fajtlowicz described in his papers on Graffiti prior to Graffiti V, the 1980s version of the program generates a collection of formulae (candidate conjectures which the user does not see), immediately after which heuristic(s) are applied by the program. As seen in Figure 7, the program generated 52 relations (correct with respect to the database), but only 23 were accepted as conjectures. Once prompted, the output was directed to the file conj.dat whose contents are seen in Figure 8. An example of a trivial candidate conjecture removed by irin may be the statement that minimum degree of the graph is not more than its maximum degree.

Before discussing the conjectures, we digress to note that the query about storing in the master file (in the lower part of the second screen shot of Figure 7) is a feature that was added in the 1990s, and thus, that feature is discussed in Section 4 of this paper. The first two options for storing or printing the conjectures are obvious. The third option, “prepare for auto”, is briefly mentioned in [25] and [28], and discussed in Section 5.1.

By the author’s count, of the 23 conjectures accepted by irin (see Figure 8 for the full list) only six were interesting. Before discussing the status of the six conjectures mentioned, we provide many relevant graph theoretical definitions.

**Definition 3.1.** Let $G$ be a simple connected graph. The *distance from vertex $u$ to vertex $v$* of $G$ is the length of a shortest $u,v$-path. The *average distance* of $G$ is the average of all distances between distinct pairs of vertices.

The *eccentricity* of a vertex $v$ of a graph is the maximum over all distances from $v$ to the other vertices of the graph. The *average eccentricity* of a graph is the average of all eccentricities of vertices of the graph. The *radius* of the graph is the minimum eccentricity and the *diameter* of the graph is the maximum eccentricity.

**Definition 3.2.** Let $G$ be a simple connected graph. A vertex of $G$ is called a *boundary vertex* if its eccentricity is maximum over all vertices. The *average distance from boundary vertices* of a graph is the average of all distances between boundary vertices and all other (distinct) vertices. A vertex of a graph is called a *center vertex* if its eccentricity is minimum over all vertices. The *average distance*
from center vertices of a graph is the average of all distances between center vertices and all other (distinct) vertices.

**Definition 3.3.** A subset of vertices of a graph is called independent if no two vertices in the subset are adjacent. The cardinality of a largest independent set is called the independence number of a graph.

The number of vertices of a largest induced bipartite subgraph of a graph is called the bipartite number of a graph. The number of vertices of a largest induced path of a graph is called the path number of a graph.

The minimum number of vertex-disjoint paths needed to cover the vertices of a graph is called the path covering number of a graph.

Among the six conjectures discussed next, one (as far as the author is aware) is open, two are known to have been previously made by Graffiti, and three are easily proven and found in mathematics papers or texts.

**Conjecture 1.** If $G$ is a simple connected graph, then the average distance from boundary vertices of $G$ is not more than independence number of $G$.

Conjecture 1 is open (as far as the author is aware).

**Conjecture 2.** [31, Conjecture labeled Graffiti 0] If $G$ is a simple connected graph, then the radius of $G$ is not more than independence number of $G$.

Conjecture 2 was proven in [40], [24], and later in [34]. At about the same time a slightly stronger result appeared in [23].

**Conjecture 3.** [31, Conjecture labeled Graffiti 2] If $G$ is a simple connected graph, then the average distance of $G$ is not more than the independence number of $G$. 
Conjecture 3 was proven in [7] (note that first a slightly weaker relation was proven in [24].) We note that the fact that irin reported both Conjecture 3 and the relation in Conjecture 1 means that the average distance from the boundary vertices and the average distance of the graph are not comparable for all simple connected graphs.

**Conjecture 4.** If \( G \) is a simple connected graph, then the diameter of \( G \) is not more than the bipartite number of \( G \).

The relation in Conjecture 4 is easily proven; it is the case that the stronger relation diameter of graph +1 is not more than the path number of graph is true (and easily proven) and mentioned in [23]. In view of this, it is reasonable to wonder why the irin heuristic made the bipartite number conjecture since indeed it follows by transitivity; however, as noted previously, this demonstration utilized a database with only 19 invariants and the path number was not among them. In Section 3.7 the database for the demonstrations is extended (and includes the path number).

**Conjecture 5.** If \( G \) is a simple connected graph, then the independence number of \( G \) is not more than the bipartite number of \( G \).

Conjecture 5 is easily proven.

**Conjecture 6.** If \( G \) is a simple connected graph, then the path covering number of \( G \) is not more than the independence number of \( G \).

This is easily proven, and noted in [52].

We note that according to Fajtlowicz the above demonstration is not typical in the sense that there can be many more conjectures accepted in one session of an execution of this version.

3.5. **Demonstration of cncl followed by irin.** For the next demonstration, we generate sums of the 19 invariants, and then utilize the ircn command, which launches the cncl heuristic followed by irin. Sums of all (distinct) invariants were generated and added to the database using the modi command (Figure 9), by choosing option 10 (to add all sums of invariants) followed by option 3 (to store...
the database). Next, the `ircn` command was given, with “simple connected graphs” selected as the property. The numeric values, 4214, 152 and 115 seen in Figure 10 are the number of relations between invariants and sums of invariants (correct with respect to the database), the number of those relations accepted by `cncl`, and the net number of relations accepted by `ircn`, respectively.

For this demonstration, we implemented a process for selecting conjectures similar to the one described in Section 3.2. First, the author categorized the conjectures as known, trivially uninteresting, and left 92 pending. Those were sent to Fajtlowicz with the request that he select some (in the spirit of what he was doing in the 1980s) that he would have considered including in _Written on the Wall_. Within thirty minutes, he responded by marking 5 with some comments; the author added
two more and sent 13 of them to an electronic mailing list\textsuperscript{7} for Fajtlowicz’s graduate students and a couple of colleagues. Note that many of the relevant definitions were given in Definitions 3.1, 3.2 and 3.3.

Among the thirteen conjectures sent to the mailing list were the 6 conjectures (with similar comments) already mentioned in Section 3.4, and the following seven conjectures.

**Conjecture 7.** If $G$ is a simple connected graph, then the diameter of $G$ is not more than the independence number of $G$ plus the radius of $G$.

The conjecture listed above was sent with Fajtlowicz’s comment, “This true because the diameter is not more than twice the radius, but the case of equality is of interest.”

**Conjecture 8.** If $G$ is a simple connected graph, then the diameter of $G$ is not more than the average eccentricity plus the radius of $G$.

Conjecture 8 was resolved shortly after the list was sent to the group. Bill Waller pointed out that this is also true, and the argument is similar to that given for Conjecture 7.

**Conjecture 9.** If $G$ is a simple connected graph, then the average eccentricity of $G$ is not more than the independence number of $G$ plus the average distance of $G$.

**Conjecture 10.** If $G$ is a simple connected graph, then the average distance of $G$ is not more than the radius of $G$ plus the minimum degree of $G$.

Shortly after the list was sent to the group, Bill Waller found a counterexample to Conjecture 10.

**Conjecture 11.** If $G$ is a simple connected graph, then the average distance of $G$ is not more than the path covering number of $G$ plus the radius of $G$.

**Conjecture 12.** If $G$ is a simple connected graph, then the minimum degree of $G$ is not more than the number of boundary vertices of $G$ plus the number of center vertices of $G$.

A couple of days after it was announced, Bill Waller and the author found a counterexample to Conjecture 12.

**Definition 3.4.** Let $S$ be a subset of the vertices of a graph $G$. Then the neighborhood of $S$, denoted $N(S)$, is the subset of vertices of $G$ that are adjacent to at least one vertex of $S$.

**Conjecture 13.** Let $G$ be a simple connected graph. Let $M$ be the subset of vertices of $G$ that have maximum degree, and let $A$ be the subset of vertices of $G$ that have minimum degree. Then the cardinality of $N(A)$ is not more than the bipartite number of $G$ plus the cardinality of $N(M)$.

3.6. **Demonstration of Beagle.** One of the commands that implement the beagle heuristic is called dirn. The command dirn proceeds similarly to the irin command except that the user is prompted for a distance (which is defined in Section 2.3). As described in Section 2.3 the convention for naming invariants affects the implementation of beagle, which utilizes the idea that conjectures involving concepts of a

\textsuperscript{7}The subject title was *A Blast from the Past.*
different type are more likely to be interesting. Since the beagle heuristic is no longer used in the Graffiti-System, the convention for naming invariants has changed. That is, Algernon no longer adheres to the convention, nor does Graffiti.pc the generator of the database (as mentioned in Section 3.1). For the demonstration, the two counterexamples were added to the database used in the previous demonstration (without sums of invariants), and the names of the 19 invariants were changed\(^8\) using a text editor.

Once the input was available, Graffiti was executed, dirn was entered as the command, and the collection of graphs for the property was selected as “simple connected graphs”. The author experimented with various values for the minimum distance between invariants, with the following outcomes. With minimum distance of 2, the beagle heuristic did not eliminate any of the 23 conjectures made by irin; with minimum distance of 3, the beagle heuristic accepted only 19 of the 23 conjectures made by irin; with minimum distance of 4, the beagle heuristic accepted 18 of the 23 conjectures made by irin; with minimum distance of 5, the beagle heuristic accepted only 10 of the 23 conjectures made by irin (Figure 11); with minimum distance of 6, the beagle heuristic accepted only 3 of the 23 conjectures made by irin; with minimum distance of 7, the beagle heuristic rejected all but one of the 23 conjectures made by irin. Finally, with minimum distance of 8, all 23 were rejected.

During the experiment, the author wondered if among the 10 conjectures (seen in Figure 12) accepted with minimum distance set to 5, one would find the six conjectures discussed in Section 3.4. It was the case that four of the six conjectures were reproduced by dirn with minimum distance set to 5. The two not present were diameter of the graph ≤ bipartite number of the graph, since according to the invariant naming convention of the 1980s version, the invariants (maximum; eccentricity; graph and order; bipartite subgraph; graph) are at distance 4; and the independence number of the graph ≤ bipartite number of the graph, since according to the old invariant naming convention, the invariants (maximum; independent vertices; graph and order; bipartite subgraph; graph) are also at distance 4.

\(^8\)For example, the min degree of graph was changed to min; degree sequence; graph.
3.7. First Demonstration of echo. The echo heuristic will be demonstrated in the next 3 sections. The first demonstration utilizes the same database as before, except that invariant names will follow the naming convention used in the 1990s. In the next two demonstrations, we expand the database to include more invariants, and illustrate two iterations of finding counterexamples and re-executing the program before listing conjectures.

The heuristic echo can be launched by itself or in combination with other heuristics. Our first demonstration will be of the command erie, which launches the irin heuristic followed by the echo heuristic. For this demonstration, as described above we utilize the same database as in the previous demonstration. The property of “triangle-free graphs” was selected as before. The program reported that 63 relations were correct (with respect to the database) for the graphs in the property selected and proceeded to prompt the user to define the background; the background selected was the collection of “simple connected graphs”. The number of conjectures accepted was reported to be three, one of which is described below.

**Definition 3.5.** Let $G$ be a simple graph. The local independence of a vertex of $G$ is the independence number of the subgraph induced by the neighbors of the vertex, and maximum of local independence of $G$ is the maximum of local independence numbers over all vertices of the graph.

**Conjecture 14.** If $G$ is a connected triangle-free graph, then the maximum degree of $G$ is not less than the maximum of local independence of $G$.

Equality of the two invariants (occurring in Conjecture 14) is trivially true for triangle-free graphs. However, this illustrates the main idea of the heuristic echo, since it is not true for all simple connected graphs. The other two reported relations were both false (and resolved by the author).

3.8. Another Demonstration of echo. For the second demonstration of erie, which implements echo, a counterexample to the previous two false conjectures was added to the database, and the number of invariants increased to 128 (although, Fajtlowicz reports using fewer than this amount). The set of invariants included
at least 8 that were computationally difficult, the others were degree and distance related. Note, the expanded database was used for the remainder of the demonstrations in this paper. Before invoking any conjecture-making commands, the modi command was utilized with option 19 selected, which deletes redundant equalities (as described in the menu of Figure 9). Option 19 prompts the user for a property. For this demonstration the property of “triangle-free graphs” was selected. The program reported 12 equalities, one of which involved the invariants in the previously discussed conjecture; most of the other equalities related some degree invariants of the graph to degree invariants of its complement. After the modi option was implemented, the database was composed of only 117 of the 128 original invariants.

Once the database was expanded, the erie command was launched, the property was selected as “triangle-free graphs” and the background was selected as “simple connected graphs”. The program reported that 1670 inequalities were generated by ineq for the selected property, and that 359 of those remained after irin was utilized. Finally, after the heuristic echo was utilized the number of accepted conjectures was 132. In Figure 13, we see that as Fajtlowicz described in [26] the program produces groups of conjectures. Since many of the conjectures on the first part of this list are false, the demonstration will include one iteration of the process (described in Section 3.2) of providing counterexamples and re-executing the program.

After considering some of the first 30 conjectures, the author found nine counterexamples\(^9\) to refute at least 13 conjectures. All refuted conjectures related degree invariants to distance invariants. Three of the counterexamples were small graphs on 6 or 7 vertices, and the others were on between 13 to 30 vertices. After adding the counterexamples to the database and re-executing, the program indicated that 1574 inequalities had been generated by ineq for the graphs of the property, and

\[^9\]We note that after the demonstration was described, Fajtlowicz commented to the author that he would have re-execute the program well before finding 9 counterexamples.
that after applying irin 342 remained. Once the heuristic echo was applied, the number of accepted conjectures was 94. The decrease in the number of conjectures is probably due to the observation that erie generates groups of similar conjectures. (That is, some of the graphs added to the database were probably counterexamples to similar types of conjectures).

After glancing at some of these 94 conjectures, the author was convinced that many of them were false. Thus, in the next section we demonstrate another iteration of the process of finding counterexamples and re-executing the program; and we also take this opportunity to discuss the idea of the touch number and to demonstrate the use of a program to test for counterexamples. Introduction and demonstration of the touch number at this juncture is intended to serve a two-fold purpose. This idea will recur in the discussion of the Dalmatian heuristic, and in the next demonstration this additional criterion (an undocumented feature of the 1980s version) might prove useful for eliminating some of the trivial conjectures made in this demonstration.

3.9. Atypical-Demonstrations of the 1980s Version. We previously noted that the demonstrations were not duplications of any of Fajtlowicz’s previous runs of this version of Graffiti; the following demonstrations diverge even more from “typical” runs of the 1980s. At this point, we will continue with the expanded database and the counterexamples found; however, we first discuss the command prox, which is not described in either the help menu or Fajtlowicz’s papers of this version, but was encountered as the author browsed the code and experimented with some commands. Note that the prox heuristic was selected (by the author) for discussion and demonstration because it involves the idea of the touch number, which is defined below and discussed again in Section 5. The touch number of a relation, $\alpha \leq \beta$, is the number of models (in the database) for which the relation is
SOME HISTORY OF THE DEVELOPMENT OF GRAFFITI

equality. The code of Graffiti indicates that the prox command launches the command ineq for a user selected property, prompts for a “proxy value” (i.e. minimum touch number) and proceeds to report which relations have a touch number of at least the “proxy value”.

For this demonstration (in an effort to continue with the line of thought in the previous demonstration), the author added a new command called erpx, which is a combination of erie and prox. The new command erpx performs exactly as erie, but first prompts the user for a minimum touch number.

Once the command erpx was entered, the minimum touch was set to 9, the property of “triangle-free graphs” was selected, and the background was selected to be “simple connected graphs”. As before, the program reported that 1574 inequalities were made by ineq for the property, 342 remained after applying irin, and 94 remained after the heuristic echo was applied; and finally, 61 relations satisfied the condition of minimum touch at least nine. Note that using the ftch command, the author had previously determined that the database contained about 90 triangle-free graphs, and thus minimum touch of 9 was arbitrarily selected at around 10% of the number of models in the property. On the first part of the list of conjectures in conj.dat as seen in Figure 15, the first two are easily shown to be correct for triangle-free graphs; however, since many of the remaining were almost certainly false, we again provide another iteration of the process of finding counterexamples, but now, rather than follow the steps of a “typical” iteration process used in the 1980s, we will use a program to test for counterexamples\(^{10}\).

The list of 61 conjectures, described by their numerical invariant identifiers, was input to Builddbs, a subprogram (described in [13]) of Graffiti.pc, for which code was rearranged to test the 61 conjectures on about 22,500 small connected triangle-free graphs; the number of vertices ranged between 1 and 16. The adjacency lists of the graphs were retrieved from The Combinatorial Object Server’s web interface for Brendan McKay’s makeg. We note that this is not the first time that conjectures of this version of Graffiti were tested on many graphs. In the early 1990s, Tony Brewster, Michael Dinneen and Vance Faber tested over 200 of Graffiti’s conjectures on all of the nonisomorphic graphs with 10 or fewer vertices; their research is summarized in [1] and [21].

Of the 61 conjectures tested by Builddbs, 42 were refuted by 25 examples of small triangle-free graphs on fewer than 11 vertices. We will present 9 of the remaining conjectures but first provide some relevant definitions common to many of the statements and note that the remaining definitions were provided previously or will precede the conjecture in which they appear.

**Definition 3.6.** Let \( G \) be a connected graph. Let \( D \) be the degree sequence of \( G \). The mode of \( D \) is the value of the sequence that occurs most frequently, and the frequency of a modal degree is the number of times a mode occurs in \( D \). We call the minimode of \( D \) the value of the sequence that occurs least frequently, and the frequency of a minimodal degree is the number of times a minimode occurs in \( D \).

Of the remaining 19 conjectures, 2 were easily proven (related to maximum degree and independence), 8 were trivially true, and finally the remaining 9 are listed next.

\(^{10}\)Fajtlowicz did experiment with a program finding counterexamples; we discuss this in Section 5.1.
**Conjecture 15.** [31, Conjecture labeled Graffiti 112] If $G$ is a connected triangle-free graph, then the radius of $G$ is not more than the frequency of a modal degree of $G$.

Conjecture 15 was disproved for triangle-free graphs by Shui-Tain Chen in April of 1988, and later she proved that the relation holds for trees.

**Conjecture 16.** If $G$ is a connected triangle-free graph, then the frequency of a minimodal degree of $G$ is not more than the 2nd smallest value in the set of degrees of the complement of $G$.

**Definition 3.7.** The closed neighborhood of a subset $S$ of vertices of $G$, denoted by $N[S]$, is defined to be the union of $N(S)$ (defined in Definition 3.4) and $S$.

**Conjecture 17.** Let $G$ be a connected triangle-free graph. Let $\bar{M}$ be the subset of vertices of $G$ that have maximum degree in the complement of $G$. Then the 2nd smallest value in the set of degrees of $G$ is not more than the cardinality of $N(\bar{M})$.

**Conjecture 18.** Let $G$ be a connected triangle-free graph. Let $\bar{M}$ be the subset of vertices of $G$ that have maximum degree in the complement of $G$. Then the maximum of even degrees of $G$ is not more than the cardinality of $N(\bar{M})$.

**Conjecture 19.** Let $G$ be a connected triangle-free graph. Let $A$ be the subset of vertices of $G$ that have minimum degree in $G$ and let $M$ be the subset of vertices of $G$ that have maximum degree in the complement of $G$. Then the cardinality of $N(A)$ is not more than the cardinality of $N[M]$.

**Definition 3.8.** Let $G$ be a connected graph, and let $S$ be a subset of the vertices of $G$. The eccentricity of $S$ is the maximum of eccentricities of the vertices of $S$. (The eccentricity of a vertex was given in Definition 3.1.)

**Conjecture 20.** If $G$ is a connected triangle-free graph, then the eccentricity of the set of boundary vertices of $G$ is not more than the 2nd smallest value in the sequence of degrees of the complement of $G$. 
Conjecture 21. If $G$ is a connected triangle-free graph, then the average distance between (distinct) center vertices of $G$ is not more than the 2nd largest value in the set of degrees of the complement of $G$.

Definition 3.9. Let $G$ be a connected graph. The length of a longest induced cycle is called the induced circumference of $G$. Note that in the case that $G$ is a tree, the induced circumference is considered by the program to be undefined (and thus trees are not considered to be counterexamples).

Conjecture 22. Let $G$ be a connected triangle-free graph. Let us denote the maximum degree of the complement of $G$ by $\Delta$. If $\Delta$ is at least two, then $n \mod \Delta$ is not more than the induced circumference of $G$.

Definition 3.10. Let $G$ be a connected graph. Let $D$ be the degree sequence of $G$ listed in non-increasing order. Let $\Delta$ be the first term of the sequence. A derived sequence is obtained from $D$ by deleting the largest element $\Delta$ and subtracting one from its $\Delta$ next largest elements; we will call this operation $L$. It is known that a sequence is realized by a graph if and only if the derived sequence is realized by a graph ([43] and [46]); thus, repeated iterations (including sorting of the derived sequences) of the operation $L$ results in a sequence of zeros. If $D$ is the degree sequence of $G$ and operation $L$ is applied until the derived sequence is the zero sequence, then the number of resulting zeros is called the residue of $G$.

Conjecture 23. If $G$ is a connected triangle-free graph, then the number of even degrees in the set of degrees of the complement of $G$ is not more than the residue of $G$.

4. The Early 1990s

In the 1989-1990 academic year, Fajtlowicz recruited graduate students to join this project as he had been awarded an Advanced Research Project grant. In the almost three years of support, the team was composed at different times of a subset of William Curry, Ermelinda DeLaVina, Siemion Fajtlowicz, Michael Granado, Kathryn Johnson and Timor Sever. First, Fajtlowicz recruited Johnson (a computer science student) to write the code for what would be called a master file, which was an effort to organize and maintain a database of Graffiti’s conjectures. The idea of the “master file” was that as the program made new conjectures one could mark their status as known, open, proven, disproven and add comments. Although the code written by Johnson ended up not being used (since the data structure used for conjectures would undergo significant changes), we note that she was the first person to contribute any code to Graffiti aside from Fajtlowicz. Further, we note that similar descriptors for conjectures were provided for in the later version; option 14 of the term menu (Figure 17) provides the ability to switch the proven status of conjectures.

In the next subsection, we describe the elementary geometry version of Algernon developed by the team of DeLaVina, Fajtlowicz, Granado and Sever. The intermediate versions of Graffiti that followed the 1980s version but preceded the Dalmatian version, which are discussed in Subsections 4.2 through 4.4 were developed by DeLaVina and Fajtlowicz (often in a joint effort).

---

[^11]: 0033652085-ARP.
[^12]: This is related to the query in Figure 7.
4.1. **Elementary Geometry.** In the middle of the spring of 1990, a main task of the team of DeLaVina, Fajtlowicz and Sever, was to write code for the subprogram *Algernon*. Specifically, the goal was to create a library module and a data module in *Algernon*, which would generate the 2-dimensional database of elementary geometry models (polygons) and invariants for input to *Graffiti* (see Figure 2 for the program structure). Although Granado joined the project later in the year, he also contributed to the elementary geometry code of *Algernon*.

Up to that point *Graffiti* had generated mathematical conjectures mainly in graph theory and some in number theory (for the latter, see conjectures numbered 434-470 and 495-536 in [31]). Since the conjecture-generating part of *Graffiti* only “knows” the values of the invariants, (and therefore does not depend on the “type” of model about which conjectures are made), Fajtlowicz’s goal to test his contention, that *Graffiti* is domain independent, seemed practical. By the summer of 1990, some geometry input was available and the program (using the 1980s version) indeed generated conjectures as expected. Eventually, the geometry conjectures were listed in [31] as numbers 726 through 744a.

In addition to *Algernon’s* new code, there were other geometry related tasks pursued. For example, the team focused for a while on triangulations of simple curves and during this time Curry (with some collaboration with Granado) wrote a Voronoi diagram viewing program. The pictures were impressive, but, unfortunately the code for the viewer was not ported to other platforms. The development of the elementary geometry databases continued through early 1991, which coincided with new versions of the conjecture-making code of *Graffiti*.

4.2. **Forever.** In 1990, Fajtlowicz began planning and later coding his ideas for enhancing the algebraic form of *Graffiti’s* conjectures. Conjectures would be inequalities between terms of an arbitrary real-based algebra. The change in the form of conjectures marked the beginning of a fundamental change in the way that the program would be developed. Obviously, the program could no longer begin by determining all correct relations relative to the database. The first application of the new capability of generating algebraic expressions, which we call terms, was in a procedure named *Forever*. The user defines a property and a background. *Forever* then systematically generates a pair of terms (algebraic expressions) subject to the beagle heuristic, evaluates the pair of terms for each model (in the database), searches for a relation between the pair of terms that holds for the graphs in the user-selected property, and applies the Echo heuristic. If the relation survives this cycle of tests, it is reported to the screen; in either case, *Forever* returns to the first step of generating a pair of terms and the cycle continues. This version’s implementation was short lived as it made too many conjectures. But this marked the end of the era in which *Graffiti’s* conjectures were only of the form $I \leq J$, $I \leq J + K$ and $I + J \leq K + L$. No conjectures generated by the *Forever* version were included in *Written on the Wall*; however, the author’s classroom notes, dated October 1990, document that Fajtlowicz presented conjectures of this version to his class. The focus of the classroom note is the following conjecture (which is correct).

---

13Given the recent successes of the educational applications of *Graffiti* we are more tempted to return to the geometry version.

14The use of the words property and background (in *Graffiti*) was given on p. 5 of this paper.
Conjecture (Forever). Let $G$ be a simple graph. The minimum degree of $G$ is not more than twice its matching number\(^{15}\).

4.3. Demonstration of Forever. The same database, as in the previous demonstration (in Section 3.9), was used as input for a demonstration of the execution of Forever. Again, triangle-free graphs were selected as the property and simple connected graphs as the background. The program’s execution was terminated (by the author) after about an hour; in that time it had generated 374 conjectures. The screen capture in Figure 16 (which is about the midpoint of the list) indicates that 95 conjectures involving the relation “less than or equal to”, and 72 conjectures involving “greater than or equal to” had been generated. Further, we note that the first three conjectures (seen in Figure 16) are false, and that the second is trivially true for triangle-free graphs. As mentioned previously, the implementation of Forever was short lived, but in the development of Graffiti it serves as a transitional moment in time from the old to the new.

4.4. Whatever. Later in 1990, Fajtlowicz had ideas for a procedure that would generate its own properties and make conjectures on the properties that it discovered. Of course, the Echo heuristic would be used. Thus, if the program made its own properties, then it should also decide on respective appropriate backgrounds. These ideas materialized as the procedure called Whatever. The planning of Whatever set the stage for many discussions on what made a property interesting.

In practice, Whatever defines new properties based on the inequalities encountered during the generation of terms. A relation between a pair of terms is considered a candidate property if it satisfies equality on a certain specified percentage of the models available in the database. However, since the program maintains a list of its discovered properties (and their respective backgrounds), it accepts a new candidate property only if it is not equivalent to any of the previously discovered properties. If the property is accepted, then the program’s list of previously discovered (in the current execution) properties and backgrounds is updated to incorporate the new property. The code indicates that there was much experimentation with

---

\(^{15}\)The matching number of a graph is the cardinality of a largest set of edges of the graph such that no two edges have a vertex in common.
assigning backgrounds. 16 During this process, the program also generates conditional conjectures for the discovered properties, using a procedure very similar to the one implemented in Forever.

After the development of Whatever, the properties generated by Whatever were also made available for use with the 1980s heuristics. This is seen in the property selection menu (Figure 5) of the 1980s version of Graffiti. Specifically, options 11 and 12 of the property selection menu allow a user to select program-generated properties for use with the heuristics of the 1980s version.

Although no conjecture of Whatever is listed in [31], this version of Graffiti is mentioned twice, once in [31] and another in [30]. In the first, Fajtlowicz included a comment above conjecture 733 in Written on the Wall, “Actually the new version of Graffiti defines properties to make conditional conjectures. In the past to get conditional conjectures, properties had to be defined by a user.” And in [30] he wrote, “...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.” Lastly, the author’s notes, dated January 1991, document a conditional conjecture (also correct) of this version, which was noted (perhaps because of its similarity to the one made by Forever, which was mentioned previously).

Conjecture (Whatever). If $G$ is a bipartite graph 17, then the minimum degree of $G$ is not more than its matching number.

Whatever made even more conjectures compared to Forever. Thus, utilization of this version was also short lived as Fajtlowicz had other ideas, 18 which we soon pursued. However, during this time, there were several technical user-oriented enhancements.

---

16 If memory serves, we ended up favoring tight backgrounds, that is, the smallest background possible. Although, the code of Whatever does not reflect this, it was evident in a procedure called Arbolito, which was a procedure in which the ideas of Dalmatian and Whatever were simultaneously implemented.

17 A bipartite graph is a graph whose vertices can be partitioned into two disjoint independent sets.

18 Upon reviewing the code, a list of problems and possible improvements noted by Fajtlowicz, served as reminders.
Since neither of *Forever* nor *Whatever* had halting conditions (as suggested by their names), it quickly became obvious that we would need a method for interrupting the program and later resuming the program. Further, since the data structures for conjectures were significantly different (than those of the 1980s version), a *ftch*-like feature, which we named *term*, was needed. The *term* option allows for viewing the list of properties discovered by the program, viewing the list of conjectures for a selected property, viewing a particular conjecture and its corresponding evaluated terms, and many other options, as can be seen in Figure 17.

Although the code for *Whatever* still exists,\(^{19}\) it is not presently executable. Thus, a demonstration in this paper for *Whatever* is not presented.

5. **The Dalmatian Heuristic**

As 1991 approached, Fajtlowicz proposed a different strategy for generating conjectures. The differences would be major. Firstly, the approach (as compared to previous versions) would be completely reversed (see Figure 18). In this version, the first heuristic applied by the program would now test for informativeness of each conjecture, and correctness would become the second consideration. In Fajtlowicz’s *On Conjectures of Graffiti V* [30] appeared the first description of *Dalmatian*, which we reproduce below.

“The program keeps track of conjectures made in the past and when it runs across a new candidate for a conjecture then first 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.”

Another major difference (not described above) was that the new approach would be driven by a larger goal. Specifically, the larger objective (as opposed to simply finding single relations) is to characterize a *fixed term*\(^{20}\) in terms of algebraic expressions involving the other invariants. That is, *Dalmatian* searches for a system of inequalities such that each inequality bounds a *fixed term*. In practice, *Dalmatian* stops if and only if for every model \(G\) in the database there exists a conjecture on the list whose *touch number*\(^{21}\) was contributed to by the model. Thus, in addition to providing a list of conjectured bounds, say for example that \(x(G) \geq c_1(G), x(G) \geq c_2(G), \ldots, x(G) \geq c_k(G)\) whenever \(G\) has property \(P\), for a user-selected *term*, \(x\), and a user-selected property, \(P\) (that defines a class of graphs), the entire list is interpreted as the following conjecture.

---

\(^{19}\)This code is still in existence, however, the program’s code as undergone multiple changes as it had to be adapted to different compilers over the years.

\(^{20}\)In Section 4.2, we described that the stream of algebraic combinations of invariants would be referred to as *terms*. By a *fixed term* we mean an algebraic combination of invariant(s) (selected by a user) that is to remain unchanged (for the duration of an execution).

\(^{21}\)This was defined in Section 3.9 as the number of models in the database for which the relation is equality.
For every model $G$ satisfying property $P$,

$$x(G) = \text{maximum of } \{c_1(G), c_2(G), \ldots, c_k(G)\}.$$  

In practice, the program generates one stream of terms (compared to two streams generated in Forever). For each instance of the stream of terms, the program first tests for “Dalmatian improvement” (see Figure 18); this is a test for determining if for at least one model (in the database) the candidate conjecture provides a closer bound (compared to previously accepted conjectures) for the fixed term selected by the user.\textsuperscript{22} If an improvement has been found, only then is the candidate conjecture tested for correctness subject to the models in the database. Further, if the program accepts the conjecture, the program then revises its list of accepted conjectures, called tree.\textsuperscript{23} If a newly accepted conjecture provides a closer bound (for all models in the database) compared to any combination of the previously accepted conjectures in tree, then a revision of the list may mean that some conjectures are moved to the program’s list of superceded conjectures, called mute. Once the list tree has been revised, the program checks if the halting condition described above is satisfied. In the program and in the diagram of Figure 18, the halting condition is called Bingo. The program reports its progress periodically, after a user specified number of iterations. Specifically, it reports its current list of accepted conjectures

\textsuperscript{22}Most often a term equivalent to an invariant is fixed; Minuteman conjectures [32], and the educational applications [6] and [54] are exceptions.

\textsuperscript{23}The accepted conjectures and removed conjectures are stored in separate structures and periodically stored to files called tree.dat and mute.dat, respectively; references to the files are seen in Figure 17.
and for each of these it reports two numerical measures (computed with respect to the database), the touch number and the average ratio\(^{24}\).

With the exception of the heuristic echo, which can be implemented by Dalmatian in the testing for Dalmatian improvement and plausibility, none of the other heuristics of the 1980s version are utilized by Dalmatian. That irin is not necessary is obvious, and instances for which encl and beagle might still be useful are rare in the author’s experience.

Before presenting a demonstration of the Dalmatian heuristic, we discuss two other developments that took place at about the same time, both of which relate to expanding the number of models accessible to the program.

5.1. Autograph and Multiple Databases. In this section, we described two early experiments of graph generation, a later development of a method for utilizing more graphs in Graffiti, and an experiment associated with these developments. In his 1989 paper, On Conjectures and Methods of Graffiti [28], Fajtlowicz mentioned a procedure called Autograph; specifically, he wrote “Autograph is a recent addition, searching for counterexamples to false conjectures.” The input for this procedure could be prepared by selecting the option “prepare for auto,” from the menu displayed in Figure 7. A search in the code of Algernon was conducted, as this Autograph described by Fajtlowicz would be a function of Algernon. Although no longer utilized\(^{25}\), the code was located (a sample is given in Figure 19). A trace of the relevant code indicates that in general the procedure searches for counterexamples to conjectures by using a given “starting graph,” and first adding or removing edges subject to the effect on conjectures. If the addition or removal of edges ceases to produce an effect on the conjectures (before a counterexample is encountered), then a new vertex is introduced to the starting graph and the process of adding or removing edges begins again. In the 1991, the name Autograph was again used for a procedure which generated graphs, which we describe next.

\(^{24}\)The average ratio is an approximate probabilistic measure of how close the values are on the left and right side of the inequality.

\(^{25}\)As described in Section 2.2 On the Code of Algernon, currently the program has procedures for inputting graphs in many forms, such as those of the programs of McKay, Brinkmann and Skiena.
In early 1991, Fajtlowicz suggested pursuing generation of graphs. A procedure named Autograph was designed, which generated graphs by applying graph operations (such as the complement or join) to a set of simple graphs. The names of derived graphs were expressed in polish notation. The generation of isomorphic graphs was, of course, a major concern. The procedure was designed to check for certain obvious redundant combinations, such as, the complement of the complement of a given graph\textsuperscript{26}, such combinations we called axioms. More complicated combinations were deduced by the procedure once the database (composed of Autograph’s graphs and various invariants which were computed for each graph) was built. The deduced redundant combinations, which we called maxioms, were output to a file and used (similarly as axioms) in subsequent re-executions of the program.

This development coincided with the development of the Graffiti-System’s utilization of multiple databases. As previously described, the programs at this time were running on the VMS/VAX Station called Charly, and extending the data structure (of the database) to accommodate more models was not possible, and thus one of the author’s tasks (at around this time) was to adapt Graffiti and Algernon to utilize and generate (respectively) multiple databases.

The graph generating procedure Autograph was adapted to create multiple databases without interrupting its execution. For the 1980s version of Graffiti, before the user is prompted for a command there is a prompt “which dbs?”, which allows a user to select the database to be used the session. For the Dalmatian version, one database is designated the “top” database on which the “Dalmatian improvement” is to be tested, and all other databases are utilized in the plausibility phase of the Dalmatian heuristic.

At about the same time the development of the procedure Autograph took place, Fajtlowicz proposed a project to analyze graphs available in the library modules of Algernon. One of the questions of interest, was how often any given graph served as a counterexample in runs of Graffiti. The interest in this was, perhaps in part, due to the issue of limited computer memory resources, but we also wanted to compare conjectures based on the Autograph database with conjectures generated by the program when using its database of counterexamples as constructed by a library module. The author’s notes indicate that after generating a database with Autograph, we set the 1980s version of Graffiti to generate conjectures implementing irin, and checked (with ftch) the correctness of conjectures with respect to another database (built by a library module) composed mainly of counterexamples. Using the latter database and again irin, the program generated about the same number of conjectures. As a result of this experiment, we concluded that counterexamples to conjectures based on the Autograph database were very special graphs such as the Peterson graph and some critical Ramsey graphs.

Another experiment with the Autograph procedure involved computing invariants recursively for graphs constructed through operations. Inspection of the code reflects that the values of many simple invariants were determined for some operators. The author’s notes indicate that a lot of time was spent considering possible algorithms, but the experiment was prematurely discontinued\textsuperscript{27}.

\textsuperscript{26}The complement of the complement of a graph is isomorphic to the graph.
\textsuperscript{27}As this occurred during the spring of 1991, there were other developments in progress and of course final exams.
5.2. Demonstration of the Dalmatian Heuristic. In 2000-2001, the author developed the program Graffiti.pc. A long-term goal for its development was to have a program similar to Graffiti on a PC platform. A short-term goal for the initial version of Graffiti.pc was to develop a program that could be used for undergraduate research; see [13] and [6] for the educational application. Graffiti.pc is a system of three programs, two of which are C++ programs Build dbs (analogous to Algernon) and Dalmatians; and the third component is a Visual Basic user interface. As suggested by the component called Dalmatians, the program’s main heuristic is the Dalmatian heuristic.

Some comparison of the two systems was first given in [13]. At this point, we discuss only the implementation of the Dalmatian heuristic, and note the following differences. There are three differences in the implementation of the Dalmatian heuristic. Two of the differences are incidental in the sense that they do not directly affect accepted conjectures (i.e. those that would appear in Graffiti’s list called tree). In Figure 20, the intermediate step “irin: transitivity” is used only for separately reporting instances of “transitivity” (i.e. cases in which a previously accepted conjecture follows from a newly accepted conjecture); we shall see an instance of this in the demonstration. Secondly, Graffiti.pc does not (at present) maintain a list of superceded conjectures (analogous to Graffiti’s mute list). The difference that may affect the list of accepted conjectures includes a user option of insisting on a minimum touch number for accepted conjectures, however, this is imposed only after the test for a “Dalmatian improvement”.

For the demonstration of the Dalmatian heuristic, the fixed term selected was the bipartite number. The choice of this term was motivated by Bill Waller’s

\[\text{Figure 20. Graffiti.pc: Dalmatian.}\]
interest in Graffiti’s conjecture number 747 in [31], which states that For G a simple connected graph, the average distance of G is not more than half the order of a bipartite subgraph. The other parameter settings for Graffiti.pc were as follows.

- The property was selected as “simple connected graphs”.
- The relation was set to greater than or equal.
- The operators used to generate terms included the 9 unary operators a half, a third, square root, multiplicative inverse, additive inverse, the floor, the ceiling, and plus or minus one; and the 3 binary operators addition, multiplication, and one operand applied as an exponent to another operand.
- The minimum touch number was set to ten.
- The “top” database was composed of most of the 128 invariants utilized in the previous demonstrations in this paper, and of about 200 graphs similar to those present in the previous demonstrations (in this paper).
- One parameter, not currently accessible through the interface, is for the utilization of multiple databases for the plausibility test of Dalmatians. For this demonstration, about 40,000 small graphs (most acquired from the Combinatorial Object Server’s web interface for Brendan McKay’s makeg) located in the multiple databases of Graffiti.pc were used.

Although, the halting condition Bingo had not been satisfied (there were about 60 graphs in the “top” database for which no conjecture on the list predicted their bipartite number), the author forwarded the following list of 18 conjectures to the mailing list mentioned in Section 3.5. Among the 18 conjectures, 16 were the conjectures accepted by Dalmatians, one was an incidental conjecture reported in the “in: transitivity” step, and another was a superceded conjecture which was forwarded to the mailing list at Bill Waller’s request; he was aware of it as the author had relayed to him an earlier list of the program’s progress. Shortly after the list was sent, the program execution was terminated. Before presenting conjectures, we introduce some notation with references to definitions.

**Notation 5.1.** Let G be a simple connected graph. We will let $b(G)$ denote the bipartite number of $G$ (Definition 3.3), $diam(G)$ will denote the diameter of $G$, $rad(G)$ will denote the radius of $G$, $ecc(G)$ will denote the average eccentricity of $G$ (Definition 3.1), and $\lambda(G)$ will denote the maximum of local independence of $G$ (Definition 3.5).

**Conjecture 24.** Let G be a simple connected graph. Then

$$b(G) \geq diam(G) + \lambda(G) - 1.$$  

**Conjecture 25.** Let G be a simple connected graph. Let $f_G(1)$ denote the frequency of degree one in the degree sequence of G. Then

$$b(G) \geq diam(G) + f_G(1) - 1.$$  

Conjectures 24 and 25 were proven by Waller and the author [17].

**Conjecture 26.** Let G be a simple connected graph. Then

$$b(G) \geq 2 \cdot rad(G).$$  

---

29 The author set the parameter in a file.

30 The use of multiple databases in Graffiti.pc is similar that described in Section 5.1.

31 This list is now part of the list Written on the Wall II, available at http://cms.dt.uh.edu/faculty/delavinae/research/wowII/ see [12].

32 It ran for about 70 hours for bipartite number. A session for path number was initiated.
Conjecture 26 was proven by Fajtlowicz in [34]; an alternate proof was given by Waller [17].

**Conjecture 27.** Let $G$ be a simple connected graph. Then
$$b(G) \geq 2 \cdot (\text{rad}(G) - 1) + \lambda(G).$$

Conjecture 27 is a superceded conjecture (removed, from the list of accepted conjectures, during the update phase described previously) included at Waller’s request; he communicated a proof for $b(G) \geq 2 \cdot \text{rad}(G) + \lambda(G) - 5$ (see [17]).

**Notation 5.2.** Let $G$ be a simple graph. We let $\alpha(G)$ denote the independence number of $G$.

**Conjecture 28.** Let $G$ be a simple connected graph. Then
$$b(G) \geq \alpha(G) + \lceil \frac{\text{diam}(G)}{3} \rceil.$$ 

**Conjecture 29.** Let $G$ be a simple connected graph. Let $M$ be the vertices of maximum degree of $G$, and let $d_{\text{max}}(M)$ be the maximum distance between vertices of the set $M$. Then
$$b(G) \geq \alpha(G) + \lceil \sqrt{d_{\text{max}}(M)} \rceil.$$ 

**Conjecture 30.** Let $G$ be a simple connected graph. Then
$$b(G) \geq \lceil \text{ecc}(G) + \lambda(G) \rceil.$$ 

**Conjecture 31.** Let $G$ be a simple connected graph. Let $d_{\text{max}}(M)$ denote the vertices of maximum degree of $G$, and let $d_{\text{max}}(M)$ be the maximum distance between vertices of the set $M$. Then
$$b(G) \geq \alpha(G) + \lceil \text{diam}(G) \rceil.$$ 

Conjecture 30 did not appear on the list of Dalmatians accepted conjectures; it is a product of the “irin: transitivity” step as described previously. Shortly after the list of conjectures was forwarded to the email list, Bill Waller and the author found a counterexample to Conjecture 30. (It is described in [17].)

**Conjecture 32.** Let $G$ be a simple connected graph. Let $B$ be the set of boundary vertices of $G$, and let $d(B, V)$ denote the average of all nonzero distances between vertices of $B$ to vertices of the graph. Then
$$b(G) \geq \lceil 2 \cdot d(B, V) \rceil.$$ 

**Conjecture 33.** Let $G$ be a simple connected graph. Let $d(V)$ denote the average of all nonzero distances between vertices of the graph $G$. Let $B$ be the set of boundary vertices of $G$, $d(B, V)$ will denote the average of all nonzero distances between vertices of $B$ to vertices of the graph. Then
$$[2 \cdot d(B, V)] \geq [2 \cdot d(V)].$$

Conjecture 33 did not appear on the list of Dalmatians accepted conjectures; it is a product of the “irin: transitivity” step as described previously. Shortly after the list of conjectures was forwarded to the email list, Bill Waller and the author found a counterexample to Conjecture 33. (It is described in [17].)

**Conjecture 34.** Let $G$ be a simple connected graph. Let $M$ be the vertices of maximum degree of $G$, and let $d(M, V)$ denote the average of all nonzero distances between vertices of $M$ to vertices of the graph. Then
$$b(G) \geq \lceil \alpha(G) + \frac{d(M, V)}{2} \rceil.$$ 

**Conjecture 35.** Let $G$ be a simple connected graph. Let $d_e(v)$ denote the number of vertices at an even distance from vertex $v$. Let $d_e(G)$ denote the minimum of $\{d_e(v) | v \in V(G)\}$. Then
$$b(G) \geq \lambda(G) + \lceil \frac{d_e(G)}{2} \rceil.$$
Conjecture 36. Let $G$ be a simple connected graph. Let $d_e(v)$ denote the number of vertices at an even distance from vertex $v$. Let $d_e(G)$ denote the minimum of $\{d_e(v) | v \in V(G)\}$. Then

$$b(G) \geq 2 \cdot \lfloor \frac{1}{2} (1 + d_e(G)) \rfloor$$

Conjecture 37. Let $G$ be a simple connected graph. Let $dd(G)$ denote the number of distinct degrees of the degree sequence of $G$. Then

$$b(G) \geq \lceil 1 + \sqrt{dd(G)} \rceil$$

Conjecture 38. Let $G$ be a simple connected graph. Let $s$ denote the minimum of $\{\lceil |N(\{u, v\})| : \{u, v\} \text{ an edge of } G \rceil \}$ for $u, v \in G$. Let $t(G)$ denote the number of subgraphs of $G$ isomorphic to a complete graph on 3 vertices. Then

$$b(G) \geq s^{t(G)}$$

Ryan Pepper communicated a proof of Conjecture 38 to the author.

Notation 5.3. Let $G$ be a simple connected graph on vertex set $V$. Let $S$ be a subset of the vertices of $G$. The minimum distance between distinct vertices of $S$ will be denoted by $d_{\min}(S)$. The maximum distance between vertices of $S$ will be denoted by $d_{\max}(S)$.

Conjecture 39. Let $G$ be a simple connected graph. Let $A$ be the subset of vertices of $G$ that have minimum degree in $G$. Let $M$ be the subset of vertices of $G$ that have maximum degree in $G$. Then

$$b(G) \geq d_{\min}(A) + \sqrt{d_{\min}(M)}$$

Conjecture 40. Let $G$ be a simple connected $n$-vertex graph. Let $A$ be the subset of vertices of $G$ that have minimum degree in $G$. Let $\Delta(G^c)$ denote the maximum degree of the complement of $G$. If $n \mod \Delta(G^c) > 0$, then

$$b(G) \geq d_{\max}(A) + \frac{1}{n \mod \Delta(G^c)}$$

Conjecture 41. Let $G$ be a simple connected graph. Let $A$ be the subset of vertices of $G$ that have minimum degree in $G$. Let $E(M^c)$ be the set of edges of the complement of $G$ induced by vertices of maximum degree in the complement of $G$. Then

$$b(G) \geq d_{\min}(A) + \sqrt{|E(M^c)|}$$

6. Closing Comments

Once the demonstrations of the 1980s version executions were described for this paper, Fajtlowicz communicated to the author one main difference not discussed in the text. Specifically, he noted that he was usually adding counterexamples one at a time (unlike the demonstrations of Sections 3.8 and 3.9) and then re-executing the program. He further noted that one can learn much by adding one counterexample at a time.33 After experiencing the compilation of demonstrations of the heuristics of the 1980s version, the author imagines that adding counterexamples one at a time may have also hinted at more of the history of the motivation for heuristics of Graffiti over the years; however, this would have only been a speculative motivation. Even so, an interesting follow-up paper might be some attempt at an analysis of each heuristic on particular types of databases.

33This seems to have inspired the Little Red Riding Hood style for implementing the program as described in [35].
Since Graffiti’s inception almost 20 years ago, in addition to the many mathematical research papers inspired by conjectures of Graffiti, the program has been discussed and compared in a variety of other papers. In 1989, an article in The New York Times [48] discussed Graffiti and its novelty as a conjecture-generating program. In the same year an inset to an article on automated theorem proving appeared in Science [8] and in 1993 an article discussing Graffiti appeared in Scientific American [47]. Over the years as computers have improved and as interest in discovery programs has increased, appropriately the nature of references to Graffiti has changed. Beginning in the late 1990s, there have been some published articles using the 1980s version of Graffiti or its conjectures for comparison and discussion; some examples are listed next.

- Pat Langley’s 1998 article, The Computer-Aided Discovery of Scientific Knowledge [49],
- Raul E. Valdes-Perez’s 1998 article, Why Some Machines do Science Well [57],
- Herbert Stoyan and Michael Müeller’s 1999 article, For the Creative, Knowledge-based Discovery of Interesting Mathematical Concepts with Methods of Artificial Intelligence [56] (Dutch),
- Simon Colton’s 2000 article, On The Notion Of Interestingness In Automated Mathematical Discovery [9],
- Pierre Hansen and Hadrien Mélot’s 2002 article, Computers and Discovery in Algebraic Graph Theory [44] and
- Pierre Hansen’s 2002 article, Computer’s in Graph Theory [45].

In addition to Fajtlowicz’s 1995 introductory description of the Dalmatian version, On conjectures of Graffiti V, other published papers that include discussion and comparison of the Dalmatian version of Graffiti include the author’s 2002 article, Graffiti.pc [13] and Craig Larson’s 2002 article, Intelligent Machinery and Mathematical Discovery [50].

As previously noted, the conjectures of Graffiti have inspired (to the present) many papers by many well known researchers; for a list of bibliographical information on papers on conjectures of Graffiti see [10]; for a list of many of Graffiti’s conjectures with some comments of Fajtlowicz see [31]. At this point, a majority of these papers are on conjectures of the 1980s version of Graffiti. Thus in closing, we provide a topical summary of the applications of the 1990s version of Graffiti with relevant conjectures, publications, preprints, and theses cited (those of which the author is aware).

- Independence number, chromatic number, length of a longest path and a chip-firing game (see conjectures numbered 747-757 in [31])
- The jet number of a graph (see conjectures numbered 778-782 in [31], also [14], [15], and [18])
- Number theory (see conjectures numbered 783-785 and 800-813 in [31])
- Ramsey r(3,a)-critical graphs (see conjectures 786-791 in [31])
- DNA sequences (see conjecture 798 and related comments in [31])
- Invariant interpolation problems (see conjectures 814-821 [31], also [38])
- k-Chromatic ramseyan properties (see conjectures 822-839 in [31], also [15] and [19])
• Chemistry (fullerenes) (see conjectures 840-862 in [31] and conjectures 895-913 in [32], also [37], [39] and [41])
• Triangle-free ramseyan properties (see conjectures 862-894 in [31], also [2], [3], [15] and [20])
• Chemistry (benzenoids) (see conjectures 914-1005 in [33])

The following applications of Graffiti are listed separately as the do not appear in [31], [32] or [33].
• Maximum number of leaves of a spanning tree (see conjectures 1-7 in [11], also [12])
• Independence number (see [11] and [16])
• Educational (see [6], [13], [35] and [54])

Acknowledgements. The author wishes to thank Inga Matthews for her interest in the project and the many suggestions, which helped improve this paper. A note of gratitude is extended to Siemion Fajtlowicz for allowing the author access to all current and old code of the two versions of Graffiti. In addition, a note of thanks is extended to the anonymous referees of this paper for their helpful suggestions.

References

[7] F. Chung, The average distance is not more than the independence number, J. Graph Theory 12 (1988), 229-235.
SOME HISTORY OF THE DEVELOPMENT OF GRAFFITI

[33] S. Fajtlowicz, Pony Express, an extension of Written on the Wall on conjectures about carcinogenic and stable benzenoids, available from Fajtlowicz.
[38] S. Fajtlowicz, Conjectures about self and acceleration of programs, manuscript, available from Fajtlowicz.

E-mail address: delavinane@uhd.edu