A Computational Attack on Graffiti's Matching and Chromatic Number Conjectures

Michael J. Dinneen

Computer Research and Applications,
Los Alamos National Laboratory,
Los Alamos, NM 87545

Abstract

In this paper we present counterexamples for over twenty of the Graffiti conjectures that deal with matching and/or chromatic number. Graffiti is a conjecture generating computer program that systematically checks for relationships among certain graph invariants. It uses a database of over 200 graphs and has generated over 700 conjectures. Our counterexamples were found by searching through all of the nonisomorphic graphs with 10 or fewer vertices. For each conjecture that failed, we display a counterexample. For those conjectures in which counterexamples seem to be very scarce, we list all that were found.

1 Introduction

This paper is a supplement to the paper [BDF] where the authors searched through a complete database of graphs with 10 or fewer vertices to find counterexamples to various conjectures of Graffiti. The results presented here deal exclusively with Graffiti's matching and chromatic number conjectures.

Graffiti is a conjecture generating program (see [Fa]) that was developed by Siemion Fajtlowicz in 1986. It operates on small databases of approximately 200 graphs and systematically checks for relationships among certain graph invariants. The Graffiti program has generated over 700 conjectures many of which have created considerable mathematical interest.

Using a readily available computer tape of all the nonisomorphic graphs with 10 or fewer vertices (see [CCRW]), we have tested roughly 60 of the Graffiti's matching and chromatic number conjectures and have found counterexamples for about one third of them. For each of the failed conjectures we present a counterexample, the values of the invariants, and the number of counterexamples.

We first give in Section 2 some general definitions and a brief discussion of how we arrived at our results. A table of passed conjectures is also given.

In Section 3 we give those conjectures for which only a small number of counterexamples were found. Next in Section 4 we present the remaining conjectures for which more than 10 counterexamples are known.

In both Sections 3 and 4 we organize the information into three categories: (1) conjectures on chromatic number, (2) conjectures on matching, and (3) conjectures on both chromatic number and matching. Also in these two sections we give table summaries of the conjectures and their counterexamples.

2 Preliminaries

2.1 Definitions

Most of the graph definitions used in this paper may be found in the glossary of [BDF] or just before their first use in [Fa]. A few terms not applicable to [BDF] are now defined.

Matching: the size of the largest matching, that is, the size of the largest subset $S \subseteq E$ of edges of a graph $G = (V, E)$ such that any vertex v in V is contained in at most one edge (a, b) of S. (Note that there may be more than one matching set S.)

perfect matching: a Matching of a graph $G = (V, E)$ that equals $\frac{|V|}{2}$ where $|V|$ is even.

chromatic number: the smallest number of colors needed to color the vertices of a graph such that two neighboring vertices must have a different color. This invariant may be abbreviated as *chromatic*.

coordinates of a set S of vertices: a vector whose i-th component is the number of neighbors of the i-th vertex in S. (Note that S is assumed to have an ordering.)

The following two prefixes are used for Matching and chromatic. If an invariant M of a graph G is preceded by the prefix **mis**, then it denotes M computed for the complement of G. Further, if an invariant M is preceded by the prefix **bi**, then it denotes $M + \text{mis}M$.

2.2 Algorithms

It is well-known that both Matching and chromatic number are in some sense computationally difficult problems. The chromatic number problem is a classic NP-complete problem, [GJ]. The Matching problem has been shown to be in P (see [Ed]), but as many programmers would say, it requires an "exponential" amount of time to implement an efficient alternating path algorithm.

Since our task was limited to graphs with 10 or fewer vertices, we devised a set of fairly efficient backtracking algorithms to find the chromatic number and the Matching of such graphs. We include these algorithms for the curious reader in Figures 1 and 2 of the appendix. C++ syntax is used in both figures.

Our chromatic number function has an additional parameter to help speed up its termination, since quite often we know the maximum clique size of the graph at hand. The first part of the algorithm is greedy in picking an initial coloration. In a systematic way, the remaining part of the algorithm does backtracking to find a best coloration.

For some of the Graffiti conjectures, we needed to know which edges yielded a maximum matching. Thus, as can be seen, additional code was included in our matching algorithm to save the largest matching (`Match M`). This backtracking algorithm is almost the same as our chromatic algorithm except that we search over all edges.

With the use of a Cray Y-MP supercomputer, our programs take approximately 6 hours (human time) to crunch through all the graphs in our database while computing both Matching and chromatic number. One may compare this time with the 2 hours needed just to read in (count) all of these same graphs.

2.3 Successful conjectures

It may be of interest to know which conjectures passed all of the graphs in our database. We briefly list these conjectures which are most likely true in Table 1. In this table we also state whether the invariants *Matching* and/or *chromatic number* were part of each conjecture. The interested reader should refer to [Fa] for the actual statements of these conjectures.

Table 1: Conjectures that passed all applicable graphs with 10 or fewer vertices.

Conjecture	Matching	Chromatic Number	Conjecture	Matching	Chromatic Number
32	no	yes	413	no	yes
119	yes	no	423	yes	no
125	no	yes	542	yes	yes
151	no	yes	550	yes	no
162	yes	no	563	yes	no
231	yes	no	572	yes	no
280	no	yes	599	yes	no
286	no	yes	604	yes	no
294	yes	no	605	yes	no
309	no	yes	618	yes	no
316	yes	no	622	yes	no
320	no	yes	624	yes	yes
323	no	yes	635	yes	yes
335	no	yes	636	yes	yes
337	no	yes	640	yes	yes
338	no	yes	645	yes	yes
339	no	yes	650	yes	yes
352	yes	yes	653	yes	yes
361	no	yes	654	yes	yes
370	no	yes	659	yes	no
371	yes	no	661	yes	no
399	no	yes	669	yes	no
412	no	yes			

3 Conjectures with 10 or Less Counterexamples

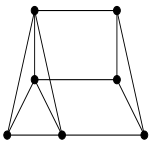
In this section we list those conjectures for which only a few counterexamples were found within our database. In some cases it may be reasonable to alter the original conjecture to exclude those few graphs that we list (or imply) below. For instance, the only counterexample we found for Conjecture 609 is the small (trivial?) complete graph of order two. Another example is Conjecture 658 where the only known counterexamples are the complete graphs of order three and greater.

Table 2 at the end of this section contains the values of the terms associated with the first counterexample given for each of the conjectures. This table is organized in the same fashion as the tables given in [BDF]. For example, referring to Conjecture 609 of Section 3.1 and Table 2 we find for K_2 that the chromatic number is two while the number of nonnegative eigenvalues is one.

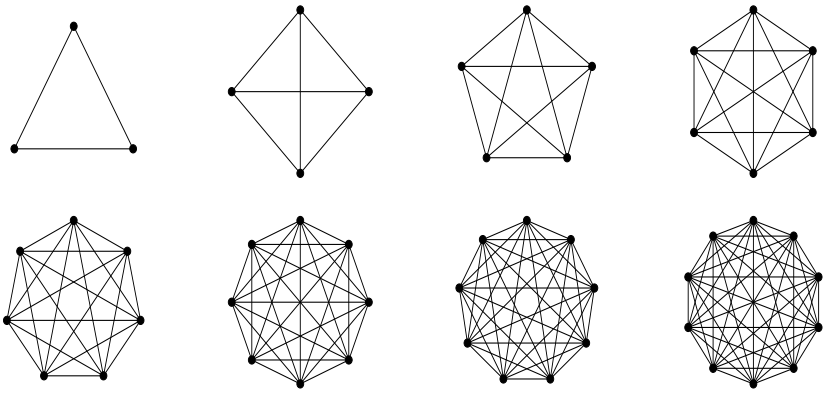
3.1 Conjectures on chromatic number

Conjecture 609: Chromatic number $\leq$ number of nonnegative eigenvalues. (connected, triangle-free graphs)

Conjecture 620: Inverse Dual Degree $\leq$ mischromatic. (graphs of diameter 2)

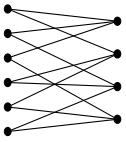


Conjecture 658: Mischromatic / independence < residue. (graphs with sum of Even $\leq$ sum of Odd)

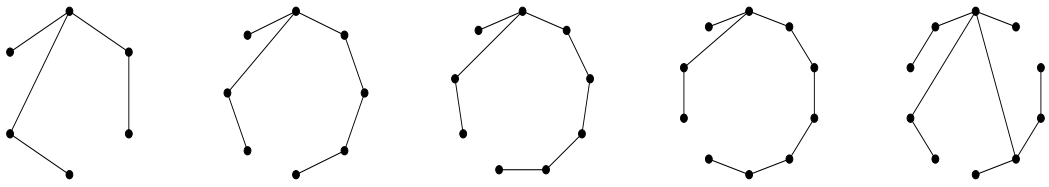


3.2 Conjectures on Matching

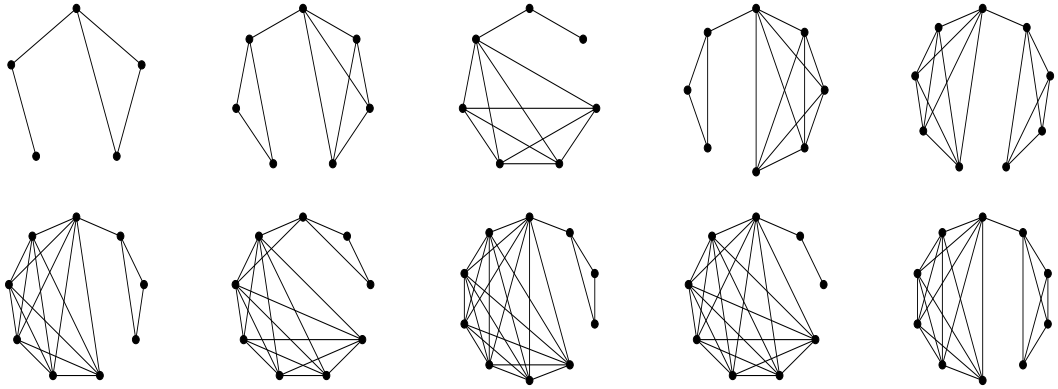
Conjecture 281: Maximum(Matching, misMatching) $\leq$ number of negative eigenvalues of Distance. (graphs with girth ≥ 5)



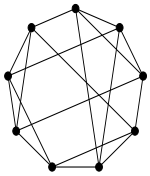
Conjecture 296: Average distance $\leq$ n / range of coordinates of matching. (trees)



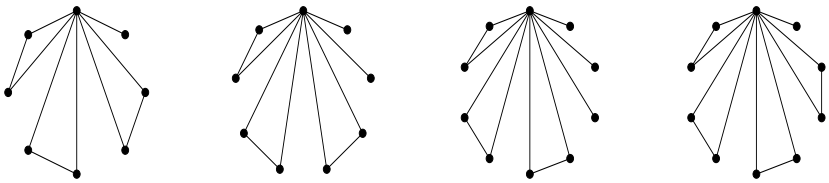
Conjecture 405: -smallest eigenvalue of Distance $\leq$ biMatching. (graphs with independence ≤ 2)



Conjecture 429: Range of positive eigenvalues $\leq$ Matching. (regular graphs)

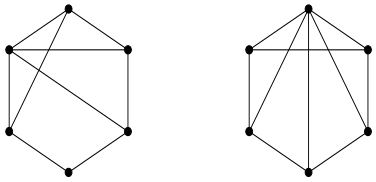


Conjecture 621: Sum of inverses of nonzero eigenvalues of Laplacian (inverse Laplacian) $\leq$ n $-$ Matching. (graphs of diameter 2)

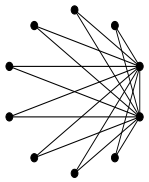


3.3 Conjectures on both chromatic number and Matching

Conjecture 637: Size / independence $\leq$ sum of positive eigenvalues. (graphs with mischromatic = n $-$ Matching)



Conjecture 652: Average distance $\leq$ inverse Dual Degree. (graphs with mischromatic = n $-$ Matching)



Conjecture 653: Average distance $\leq$ frequency of mode of 1-Residue. (graphs with mischromatic = n $-$ Matching)

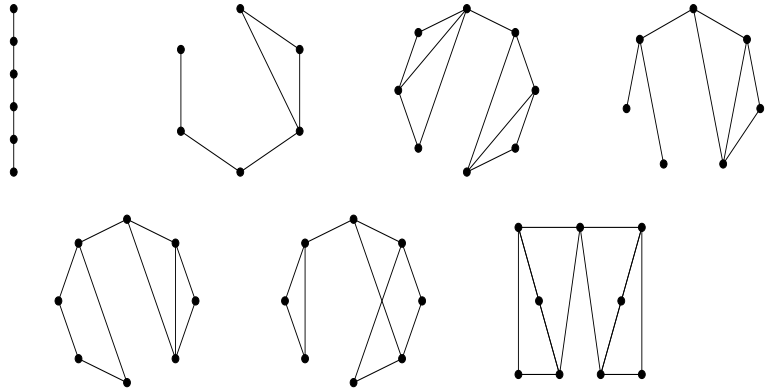


Table 2. The Value of the Terms Associated with the First Counterexample: Conjectures with 10 or Less Counterexamples

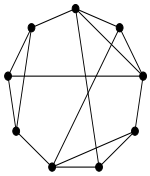
Conjecture	Number of counterexamples	1st term	2nd term	3rd term	4th term	5th term
281	1	5	4	6	-	-
296	5	2.0667	6	3	-	-
405	10	4.2626	4	2	-	-
429	1	5	4	-	-	-
609	1	2	1	-	-	-
620	1	2.0071	2	-	-	-
621	4	4.1125	8	4	-	-
637	2	9	2	4.4788	3	-
652	1	1.6222	1.6089	8	-	-
653	7	2.3333	2	3	-	-
658	8	1	1	1	3	6

4 Conjectures that have More than 10 Counterexamples

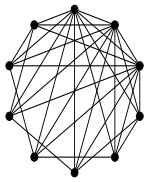
This section presents those conjectures for which we have found more than 10 counterexamples. Since there are an abundant number of graphs for these conjectures, a pictorial view is given for only the first counterexample found. Table 3 contains the values of the terms associated with those graphs presented along with the total number of counterexamples found.

4.1 Conjectures on chromatic number

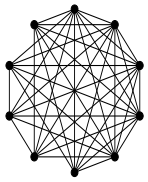
Conjecture 264: 2 + range of positive eigenvalues $\leq$ bichromatic.



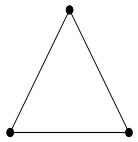
Conjecture 381: Mischromatic $\leq$ number of nonnegative eigenvalues. (graphs of diameter 2)



Conjecture 574: Mischromatic / independence $\leq$ mode of Even. (connected graphs)

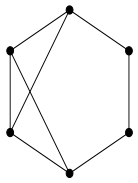


Conjecture 660: Mischromatic / independence < range of positive eigenvalues. (graphs with sum of Even ≤ sum of Odd)

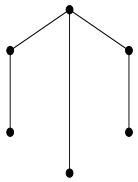


4.2 Conjectures on Matching

Conjecture 188: Mode of eigenvalues of Laplacian ≤ n − Matching. (graphs with sum of Odd ≤ sum of Even)

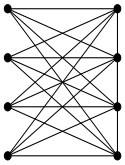


Conjecture 598: Range of coordinates of matching ≤ n / average distance. (triangle-free graphs)

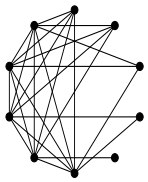


4.3 Conjectures on both chromatic number and Matching

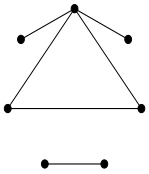
Conjecture 340: 2 + minimum positive eigenvalue ≤ bichromatic. (graphs with a perfect matching)



Conjecture 644: Minimum of Dual Degree $\leq$ mean of Odd. (graphs with mischromatic = n − Matching)



Conjecture 646: Randic $\leq$ maximum frequency of coordinates of a maximum clique. (graphs with mischromatic = n − Matching)



Conjecture 649: There are graphs with mischromatic / independence $\geq$ range of positive eigenvalues. (graphs with mischromatic = n − Matching)

Conjecture 651: Average distance $\leq$ maximal frequency of Degree. (graphs with mischromatic = n − Matching)

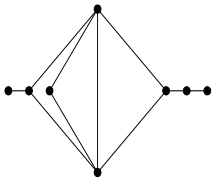


Table 3. The Value of the Terms Associated with the First Counterexample: Conjectures with More than 10 Counterexamples

Conjecture	Number of counterexamples	1st term	2nd term	3rd term	4th term	5th term
188	67713	4	6	3	16	18
264	582	5	6	-	-	-
340	29	5.123106	7	-	-	-
381	33	4	3	-	-	-
574	15	3	2	1	-	-
598	160	3	6	2.0667	-	-
644	246	5.4286	5.4000	5	-	-
646	444994	3.207107	3	3	-	-
649	73	1	1	1	1	-
651	715	2.1786	2	4	-	-
660	105	1	1	2	3	6

References

[BDF] T.L. Brewster, M.J. Dinneen, and V. Faber, *A Computational Attack on the Conjectures of Graffiti: New Counterexamples and Proofs*, January, 1992.

[CCRW] R.D. Cameron, C.J. Colbourn, R.C. Read, N.C. Wormald, *Cataloguing the Graphs on 10 Vertices*, Journal of Graph Theory, **9** (1985) p. 551-562.

[Tape announcement, *Discrete Math.* **31** (1980) p. 224.]

[Ed] Jack Edmonds, *Paths, Trees, and Flowers*, Canadian Journal of Mathematics, **17** (1965) p. 449-67.

[Fa] Siemion Fajtlowicz, *Written on the Wall*, Department of Mathematics, University of Houston, February 19, 1987, Updated May 26, 1991.

[GJ] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, San Francisco, CA, 1979.

Appendix

Figure 1: A backtracking algorithm used to find the chromatic number.

```
/*
 * Chromatic number of graph.
 */
int chromatic(const Graph &G, int lower_bound)
 {
     int n = G.order();

     short int vert[n];         /* Current assignment of colors. */
     int max_count=1;           /* Number of colors used. */
     int i, j, c;
     int first_max;
     int direction;

   /*
    * Be greedy to find an initial coloration.
    */
   for (i=0; i<n; i++)
    {
     for (c=0; c<max_count; c++)
       {
         for (j=0; j<i; j++)
            if (G.is_edge(i,j)==true && vert[j]==c) goto Next_c;

         vert[i]=c;
         goto Next_i;
Next_c: ;
       }
     vert[i]=max_count;
     max_count++;
Next_i: ;
    }
```

Figure 1 (continued): A backtracking algorithm used to find the chromatic number.

```
   /*
    * Try to use fewer colors.
    */
Lower_max:
   if (max_count==lower_bound) return max_count;

   for (first_max = --max_count; vert[first_max]!=max_count; first_max++);

   for (i=first_max-1,direction=0; direction ? i<n : i>=0; i+=direction-1)
    {
     if (vert[i]+1 < max_count+direction)
       for (c = (direction ? 0 : vert[i]+1); c<max_count; c++)
        {
         for (j=0; j<i; j++)
           if (G.is_edge(i,j)==true && vert[j]==c) goto Next_c2;

         vert[i]=c;
         direction=2;
         goto Next_i2;
Next_c2: ;
        }
         direction=0;
Next_i2: ;
    }

   if (i<0) return max_count+1;
   else     goto Lower_max;
 }
```

Figure 2: A backtracking algorithm used to find a maximum matching.

```
/*
 * Maximum Matching of a graph.
 */
int matching(const Graph &G, Match &M)
 {
   int n = G.order();

   short int vert[n];         /* vert[i] is matched up with match[i] */
   short int match[n];
   Bool used[n];
   int current_match=0;
   int isolated=0;
   int max_match=0;
   int best_match=n/2;
   int i, j, m;

   for (i=0; i<n; i++) used[i]=false;

   /*
    * Be greedy to find a matching.
    */
Add_matching:
   for (i = current_match ? vert[current_match-1]+1 : 0; i < n-1; i++)
    {
     if (used[i]) continue;

     for (m=i+1; m<n; m++)
       {
         if (used[m]) continue;

         if (G.is_edge(m,i))
          {
           vert[current_match]=i;
           match[current_match]=m;
           used[m]=true;
           current_match++;
           break;
          }
       }
    }
```

Figure 2 (continued): A backtracking algorithm used to find a maximum matching.

```
if (max_match < current_match-isolated)
 {
   max_match=current_match-isolated;
   M.save_match(vert,match);
 }

if (current_match-isolated == best_match) return best_match;

/*
 * Backtrack phase to try and improve matching.
 */
for (j=current_match-1; j>=0; j--)
 {
  if (match[j] == n) isolated--;
  else
  if (match[j] == n-1)
   {
     used[match[j]]=false;
     match[j]=n;
     isolated++;
     current_match=j+1;
     goto Add_matching;
   }
  else
   {
     used[match[j]]=false;

     /*
      * Try to just alter current match.
      */
     for (m=match[j]+1; m<n; m++)
      {
      if (used[m]) continue;

      if (G.is_edge(m,vert[j]))
       {
        used[m]=true;
        match[j]=m;
        current_match=j+1;
        goto Add_matching;
       }
      }
```

Figure 2 (continued): A backtracking algorithm used to find a maximum matching.

```
      /*
       * See if we can replace this match with a later one.
       */
      for (i = vert[j]+1; i<n-1; i++)
       {
        if (used[i]) continue;

        for (m=i+1; m<n; m++)
          {
            if (used[m]) continue;

            if (G.is_edge(m,i))
             {
              vert[j]=i;
              used[m]=true;
              match[j]=m;

              current_match=j+1;
              goto Add_matching;
             }
          }
       }

      /*
       * All of the above failed so backtrack. (next j)
       */
     }
   }

 return max_match;
}
```