

How to Color a Graph with (Computer) Algebra

Yuri Matiyasevich

*Steklov Institute of Mathematics at Saint-Petersburg
27 Fontanka, Saint-Petersburg, 191011, Russia
E-mail: yumat@pdmi.ras.ru
URL: <http://logic.pdmi.ras.ru/~yumat>*

This note presents some nonevident ways for determining the colorability of a graph and the number of its colorings via algebraic manipulations of polynomials in many variables.

Suppose we have a graph G and are interested in learning whether one can properly color its vertices in k colors. In other words, we want to know whether one can assign a color $c[i]$ to the i -th vertex in such a way that the ends of each edge of G would get different colors, $c[i]$ belonging to some k -element set $\mathbf{Colors}[k]$. *Could we find the answer with general computer algebra systems?*

On one hand, such systems are universal in the sense that they can compute everything that can be computed in principle, so the answer is definitely positive.

On the other hand, computer algebra systems were initially intended for dealing with mathematical formulas, so the required program might be rather cumbersome. To make our problem more challenging, let us state it in the following way:

Q: Write the shortest possible program `ColorableQ` which would determine, for a given graph G and a natural number k , whether graph G is vertex colorable in k colors.

Pay attention that we do not ask for the *most efficient* program; in fact, the intrinsic computational complexity of graph coloring is not yet known.

The solution of our problem depends, of course, on the particular system exploited. A specialized system can contain **ColorableQ** as a standard function, making the solution of our problem trivial. Using the *Mathematica* standard add-on package *Combinatorica*, one can easily define **ColorableQ** as follows.

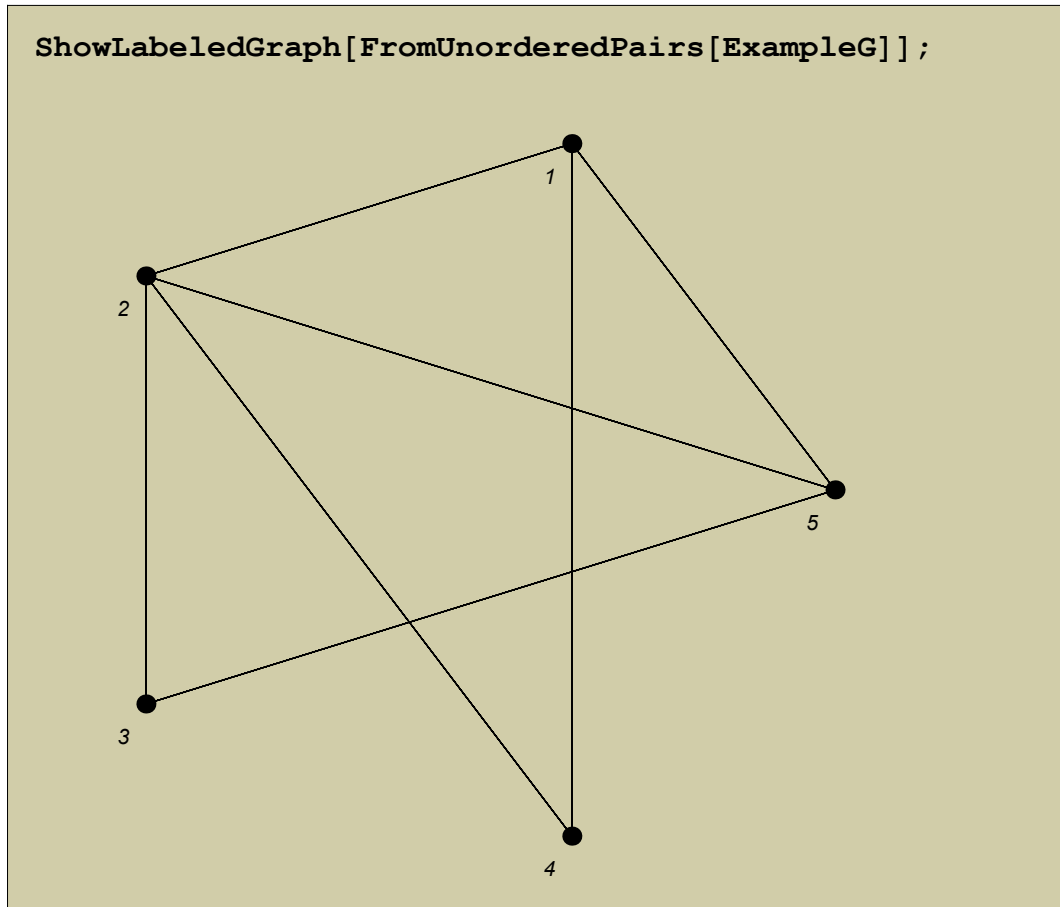
```
<< DiscreteMath`Combinatorica`
```

```
ColorableQ[G_, k_] := ChromaticPolynomial[G, k] > 0
```

Thus, in order to keep our problem challenging, we need to restrict the admissible tools to basic algebraic manipulations with formulas.

Another point to be taken into account is the representation of graphs. *Combinatorica* uses an adjacency matrix in its canonical representation for graphs. However, it also allows a more compact representation via an unordered set of adjacent vertices. We will be looking for a solution to our problem with this alternative representation.

```
ExampleG = {{1, 2}, {2, 3},  
            {3, 5}, {1, 4}, {2, 4}, {1, 5}, {2, 5}};
```



Here is one possible program. Is it the shortest one based on “first principles” only?

```
ColorableQ[G_, k_] := Module[{c, d, i},
  (Expand[Apply[Times, Apply[Subtract, Map[c, G, {2}],
    1]]) /. c[i_]^d_ -> c[i]^(d mod k) != 0]
```

```
ColorableQ[ExampleG, 2]
```

```
False
```

```
ColorableQ[ExampleG, 3]
```

```
True
```

Let us see why it works. To this end we first rewrite the program in a lengthy form with notation for the intermediate objects.

```
ColorableQ[G_, k_] := (
  Clear[c];
  P1 = Map[c, G, {2}];
  P2 = Apply[Subtract, P1, 1];
  P3 = Apply[Times, P2];
  P4 = Expand[P3];
  DegreeReduction = c[i_]^d_ -> c[i]^d mod k;
  P5 = P4 /. DegreeReduction;
  P5 != 0)
```

The **Map** (applied at level 2) transforms **G**, a list of numbers of adjacent vertices, into a list of pairs of colors which should be different.

```
ColorableQ[ExampleG, 3];
P1
{{c[1], c[2]}, {c[2], c[3]}, {c[3], c[5]}, {c[1], c[4]},
 {c[2], c[4]}, {c[1], c[5]}, {c[2], c[5]}}
```

The first **Apply** produces a list of differences of colors which should all be different.

P2

$$\{c[1] - c[2], c[2] - c[3], c[3] - c[5], \\ c[1] - c[4], c[2] - c[4], c[1] - c[5], c[2] - c[5]\}$$

As soon as we replace all $c[i]$ by some values (from our k -element set **Colors**[k]) that do *not* produce a coloring of the graph G , the list **P2** will contain a zero.

The second **Apply** transforms **P2** into a polynomial.

P3

$$(c[1] - c[2]) (c[2] - c[3]) (c[1] - c[4]) \\ (c[2] - c[4]) (c[1] - c[5]) (c[2] - c[5]) (c[3] - c[5])$$

This polynomial is equal to zero for any choice of values of $c[i]$ from $\mathbf{Colors}[k]$ that is not a coloring of the graph.

P4 is another representation for the polynomial **P3**, so it is equal to zero for each choice of values of $c[i]$ from the set $\mathbf{Colors}[k]$ if and only if the graph G has no colorings in k colors. The size of **P4** is exponentially greater than the size of **P3**, and this is the point that makes our program computationally nonefficient.

The substitution **DegreeReduction** is the heart of the program. In graph theory, colors are not specified. We are free to select for $\mathbf{Colors}[k]$ any set with k elements, and our choice is the set of all k -th roots of unity.

```
Colors[k_] := Table[Root[ck - 1, d], {d, 1, k}]
```

Under this interpretation of colors, the left-hand side of **DegreeReduction** is equal to its right-hand side, so the values of the polynomials **P4** and **P5** coincide as long as the values of $c[i]$ are taken from the set $\mathbf{Colors}[k]$.

We saw that if the graph G has no colorings in k colors, then both polynomials **P4** and **P5** are equal to zero for each choice of values of $c[i]$ from the set $\mathbf{Colors}[k]$. But the polynomial **P5** has degree at most $k-1$ in each variable, and by the interpolation theorem there is only one such polynomial. In other words, the graph G has no colorings in k colors if and only if the polynomial **P5** is *identical* to the zero polynomial, and that is what is checked in the last line of the program.

Thus, there is no magic behind **ColorableQ**. But what is required if we ask about the number of colorings rather than about their mere existence?

Q: *Using only basic tools for formula manipulation, write the shortest possible program **NumberOfColorings** which*

would determine, for a given graph G and a natural number k , the number of vertex colorings of the graph G in k colors.

It turns out that a slight modification of **ColorableQ** enables us to solve this more difficult problem. First of all, we need to replace the standard **Subtract** by a special function.

```
ColorSubtract[c1_, c2_] :=
  1 - Sum[c1d c2k-d, {d, 1, k}] / k
```

Similar to **Subtract**, the value of **ColorSubtract**[**c1**, **c2**] is equal to 0 as soon as **c1** and **c2** are equal elements from **Colors**[**k**]. However, if **c1** and **c2** are unequal elements of this set, then the value of **ColorSubtract**[**c1**, **c2**] is equal to 1, because in this case the sum is equal to $(c1^k - c2^k) / (c1 - c2)$.

Modifying the final part of the former program, we get the desired new one.

```
NumberOfColorings[G_, k_] := (
  Clear[c];
  P1 = Map[c, G, {2}];
  ColorSubtract[c1_, c2_] :=
    1 - Sum[c1d c2k-d, {d, 1, k}] / k;
  P2 = Apply[ColorSubtract, P1, 1];
  P3 = Apply[Times, P2];
  P4 = Expand[P3];
  DegreeReduction = c[i_] d . => c[i] d mod k;
  P5 = P4 /. DegreeReduction;
  P6 = P5 /. c[i_] -> 0;
  P6 * kV[FromUnorderedPairs[G]])
```

P6 is just the free term of **P5**. It turns out to be equal to the number of colorings divided by $k^{V[G]}$, where $V[G]$ is the number of vertices of graph G . Why?

Due to the fact that the value of **ColorSubtract** is either 1 or 0, the value of **P4** and **P5** is equal to 1 or 0 depending on whether the values of **c**[**i**] produce a

coloring or not. Having a small degree in each variable, the polynomial $\mathbf{P5}$ can again be restored from these values by the interpolation theorem. The Lagrange form of the interpolating polynomial would consist of exactly $\mathbf{NumberOfColorings}[G, k]$ summands, and it is not difficult to check that the free term of each summand is equal to $k^{-V[G]}$.

Removing redundant notation, we get the following short form of $\mathbf{NumberOfColorings}$.

```
NumberOfColorings[G_, k_] :=
Module[{c, d, i}, (Expand[Apply[Times,
  Apply[1 - Sum[#1d #2k-d, {d, 1, k}] / k &,
    Map[c, G, {2}], 1]]] /. c[i_]d -> c[i]d mod k /.
  c[i_] -> 0) * kV[FromUnorderedPairs[G]]
```

```
NumberOfColorings[ExampleG, 4]
```

```
96
```

The latter output can be compared with

```
ChromaticPolynomial[FromUnorderedPairs[ExampleG], 4]
```

96

In the author's HTML paper, <http://logic.pdmi.ras.ru/~yumat/Journal/Triangular/triang.htm>, one can find more nonevident algebraic ways for calculating the number of colorings of plane triangulations, giving new restatements of the Four-Color Theorem.

About the Author

Yuri Matiyasevich is the head of the Laboratory of Mathematical Logic at Steklov Institute of Mathematics in St. Petersburg, Russia. His most known achievement was the negative solution of Hilbert's tenth problem (<http://logic.pdmi.ras.ru/Hilbert10/stat/stat-eng.html>) about which he wrote a book (<http://logic.pdmi.ras.ru/~yumat/H10Pbook/index.html>, English translation published by the MIT Press in 1993). Besides logic, Yuri Matiyasevich works in number theory and discrete mathematics.