Practice 3 report

Guillermo Girón García

December 6, 2019

Index

Abstract

Introduction

Methodology

Results and discussions

Conclusions

References

Abstract

This pages are dedicated to the resolution of a Petersen graph colouring, reducing it to a SAT problem, for two colours, being impossible to have a solution and for three and four colours, which both have solutions.

Introduction

In this practice we're going to analyze the possibilities to solve a Petersen graph colouring in several number of colours through a reduction made to SAT problem, introducing for this purpose, the dimacs sintax and picosat software.

We're going to learn too about Petersen graphs properties, its chromatic number, its clique number, and some differente colorabilty possibilities.

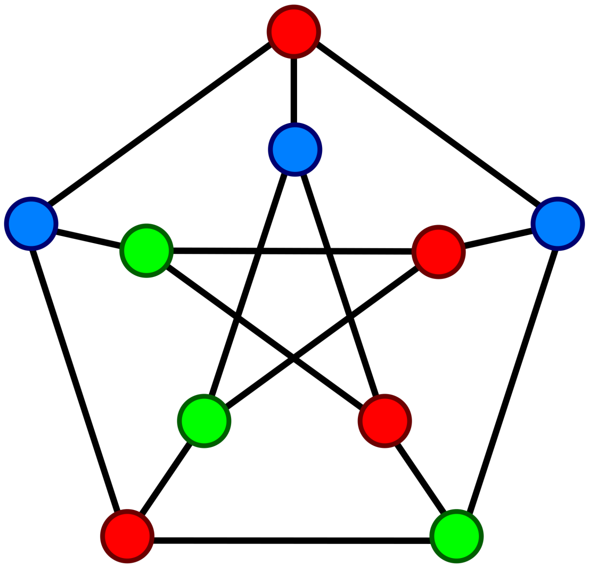


Figure 1: 3-coloured Petersen graph

Methodology

The only physical equipment needed for this experiments, it's a computer with some Linux distribution installed or a Virtual machine to use the requested software: PicoSat.

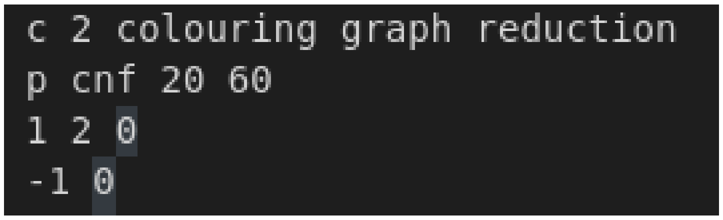
We'll proceed, learning about dimacs and its semantics. In this pseudo language, we compute a problem decomposed in clauses. It's similar to a rule-based system.

The first line starts with letter 'c' and represents the comment line, to clarify what the algorithm does.

The second one, starts with letter 'p', from problem, followed by the type of problem, in our case 'cnf', then the number of variables we'll use and the number of clauses we'll have in our problem. Both of them, we'll depend on wether we're computing 3 or 2 graph colouring.

In dimacs, every clause it's formed by variables separated by a blank space, and every clause it's finished in a 0. Every variable it's an integer number, that can be negative if the variable is negated.

This variables, represents the possibles colour adopted for every node in our graph. Figure 2 shows an example.



```
c 2 colouring graph reduction
p cnf 20 60
1 2 0
-1 0
```

Figure 2: Dimacs example extracted from 2 colour reduction

We create two c++ programs that computes the translation from a graph to dimacs, given it's adjacency list in a file called list.txt. One of them will compute 3 graph colouring and another one 2 graph colouring.

Then, this dimacs translation are passed through another software called PicoSat, that gives the answer on wether the problem have a valid solution or not.

From now on, we'll call 'k', to the number of colours during our reduction, and 'n', to the number of vertex in our graph. So our number of variables for reduction will be k * n.

Let's focus now on how to build clauses for each vertex.

- There will be a clause for node, forcing it to take a colour. Taking as example 3 graph colouring, we can have: $R_i v B_i v G_i$
- For every node, there will be a number of clauses to prevent it from having more than one colour. This means we need all the 2-permutations from the k colours. Ex: $(\neg R_i \, v \, \neg B_i) \, and \, (\neg R_i \, v \, \neg G_i) \, and \, (\neg B_i \, v \, \neg G_i)$

The amount of 2-permutations for k colours is an arithmetic progression going from 1 to k-1, with a difference of 1. It can be computed by the next formula:

$$\frac{k*(k-1)}{2}$$

So, for now, we have one, plus the formula above clauses, for every node.

- Lastly, we have to set the constraints that will prevent two adjacent nodes from having the same color. To do that, we need to run through every node's adjacency list and add k clauses per every adjacent node. Ex: $(\neg R_i v \neg R_j)\, and\, (\neg B_i v \neg B_j)\, and\, (\neg G_i v \neg G_j)$

Since adjacency lists can have any number of vertices up to n, there is no formula to calculate the number of clauses needed. So we have to add up every vertex's adjacency list's size times k.

However, the graphs we are working with are undirected graphs, which means that every adjacency is listed twice, so we have to divide the previous amount of clauses by two.

In order to avoid constructing duplicated clauses, I simply add a restriction when reading a vertex' adjacency list: the clause is only written if the "origin" vertex (the one whose list we are reading) is lesser than the "destiny" vertex (the one inside the list). This way, we would write a clause if we find the vertex 7 in 2's list, but not if we find the vertex 2 in 7's list, for example.

With that said, the total amount of clauses would follow the formula:

$$n + n*\frac{k*(k-1)}{2} + edges*k$$

Where, of course, we don't have the number of edges, and instead have to check the adjacency lists.

Results and discussions

After testing the programs, one can easily see that the Petersen graph is unsolvable with two colors, as both the graphColor function and the PICOSAT application fail to find a solution. The graph is however colorable with three colors, with 120 different possible solutions, and 12960 solutions with 4 colors. With this, we can say that the chromatic number of the Petersen graph is three.

Two of the possible 3-color solutions would be the following (assuming that 1, 2 and 3 are different colors, and the vertices are called in order from 1 to 10):

1 2 1 2 3 2 1 3 3 2
2 3 2 3 1 3 2 1 1 3

Conclusions

First of all, we have successfully made a Karp-reduction from graph coloring to SAT problems, as the reduction has no instructions that take longer than polynomial time, and the SAT-solving algorithm only has to be called once to obtain an answer.

We have also proved that the Petersen Graph's chromatic number is three, as it is not solvable with two colors.

If the graph is three-colorable, that means that if we pick any four vertices of the graph, at least one color will be repeated. If that is the case, then it is not possible for said four vertices to form a clique, as they would not satisfy the graph coloring problem. As such, it is impossible for the Petersen graph to have a clique of size four.

In fact, the maximum possible clique that any graph can have is its chromatic number, if we apply that same logic to any other case. This does not mean the clique will be of that size, only that it can't be bigger. In Petersen's case, its maximum clique is of size two, as it doesn't form any triangles.