Assignment 3. Polynomial Reductions

María Segura Bolaños

Computational Complexity
School of Engineering
University of Cádiz

maria.segurabolanos@alum.uca.es

11th December 2023

Summary

In this Computational Complexity practice, we are focus on the Clique problem in its three modalities: decision, search and optimization. For the decision version, we get the solution by reducing the problem to a the Satisfiability problem. In the search problem, we are going to use the decision version as an oracle, by removing all the nodes from the graph one by one and checking if the clique continues in the graph or not, that is, if the eliminated node belonged to the clique or not. Finally, in the optimization version, we start by searching the size of the maximum clique by using the binary search with the decision problem, and once we know that value, we call the search function to get the nodes that make up the clique.

Contents

1 The clique problem on its decision version **2**

2 The clique problem on its search version **4**

3 The clique problem on its optimization version **4**

Tables

Figures

1 Graph 3
2 Decision problem 3
3 Solution checked with cliquer 3
4 Search problem 4
5 Optimization problem 4

1 The clique problem on its decision version

We define a clique as a set of nodes in which each pair of nodes is connected by an edge, thus forming a complete subgraph. On its decision version, we are asked if given a k such that 1 ≤ k ≤ number of vertices of the graph, does the graph contains a k-clique. In this case, I have reduced it to the SAT problem.

In the program, the graph is given with a txt that uses the DIMACS format for graph, this means that the first line contains "p edge n m", with n equals to the number of vertices and m equals to the number of edges, and the rest of the lines of the document are like "e v u", being v and u the two nodes that are connected.

Then, in the file data.hpp, I have developed a function that generates a txt with all the clauses that make up the clique problem when it is reduced to the Satisfiability problem. These clauses are divided in three parts:

1. $x_{1r} \vee x_{2r} \vee ... \vee x_{nr}$ for all $1 \leq r \leq k$. It wants to say that at least one node of the graph represents the r node of the clique if the SAT problem is satisfiable.
2. $\neg x_{ir} \vee \neg x_{is}$ for all $1 \leq i \leq n$ and $1 \leq r < s \leq k$. It is that a node of the graph that belongs to the clique, just can be once in the clique.
3. $\neg x_{ir} \vee \neg x_{js}$ for all $1 \leq i, j \leq k$ such that $(i, j) \notin E$ and $r \neq s$. This means that if there is no edge between two nodes in the original graph, at least one of the nodes doesn't belong to the clique.

I have done a Cook reduction: I have transformed the instance given of the Clique problem to the SAT problem in a polynomial time, and by getting the solution of the Satisfiability problem, I can answer to the original problem. This is important because thanks to this, we can proof that both problems belong to the class NP-Complete.

We can proof the order of the reduction because the number of clauses generated always is:

1. k clauses
2. n * ((k - 1) * k) / 2 clauses
3. ((n * (n - 1) / 2) - nEdges) * k * (k - 1) clauses

with n equals to the number of vertices of the graph, k equals to the size of the clique and nEdges the number of edges that have the graph.

If the program is compiled (g++ decision.cpp -o "decision") and executed (./decision), we can check that there is at least one 4-clique, but there isn't any 5-clique in the following graph.

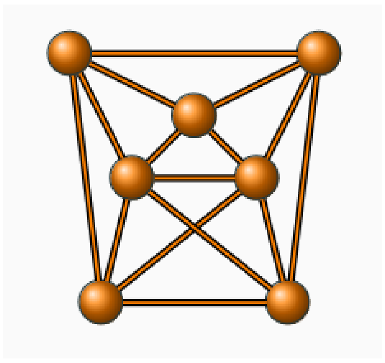
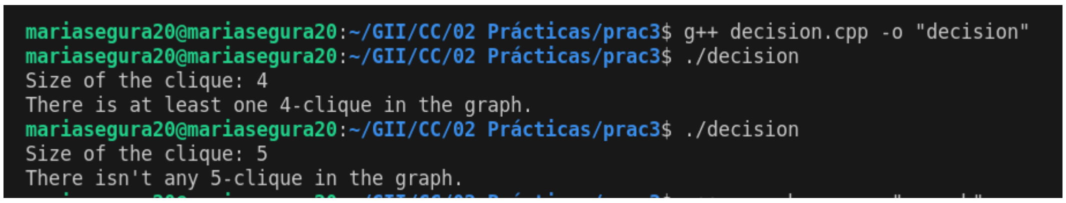


Figure 1: Graph

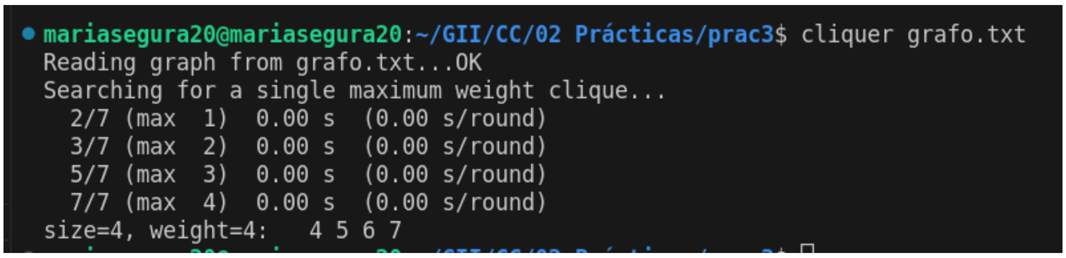


```
mariasegura20@mariasegura20:~/GII/CC/02 Prácticas/prac3$ g++ decision.cpp -o "decision"
mariasegura20@mariasegura20:~/GII/CC/02 Prácticas/prac3$ ./decision
Size of the clique: 4
There is at least one 4-clique in the graph.
mariasegura20@mariasegura20:~/GII/CC/02 Prácticas/prac3$ ./decision
Size of the clique: 5
There isn't any 5-clique in the graph.
```

Figure 2: Decision problem

The program has been developed by using PICOSAT, which recieve the file with all the clauses generated and return SATISFIABLE or UNSATISFIABLE as the answer to the SAT problem, what is translated in true or false to the question of the clique problem on its decision version.

By using the clicker command followed by the grafo.txt file, we can check that the results are right:



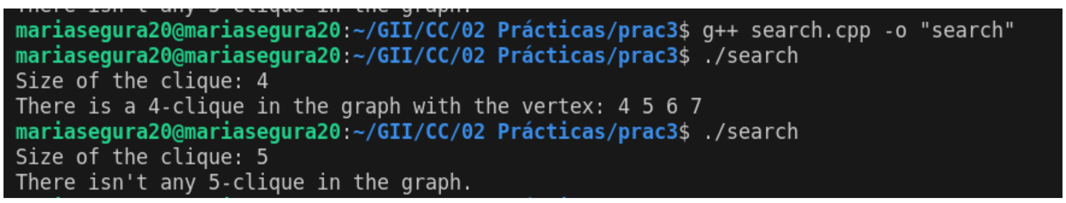
```
● mariasegura20@mariasegura20:~/GII/CC/02 Prácticas/prac3$ cliquer grafo.txt
  Reading graph from grafo.txt...OK
  Searching for a single maximum weight clique...
    2/7 (max  1)  0.00 s  (0.00 s/round)
    3/7 (max  2)  0.00 s  (0.00 s/round)
    5/7 (max  3)  0.00 s  (0.00 s/round)
    7/7 (max  4)  0.00 s  (0.00 s/round)
  size=4, weight=4:   4 5 6 7
```

Figure 3: Solution checked with cliquer

It's important to say to know which nodes say the cliquer that make up the 4-clique, that in the file "grafo.txt", the nodes have been named from left to right and from top to bottom starting with one.

2 The clique problem on its search version

The search version of the Clique problem is that, given $1 \leq k \leq n$, find a k-clique of the graph, if any. I have developed a function to solve this (in search.hpp) which go through all the nodes by removing it, calling the decision version, and if the answer is yes, it means that the processed node doesn't belong to the clique, and if the answer is no, it means that the processed node belong to the clique so it must be restored.



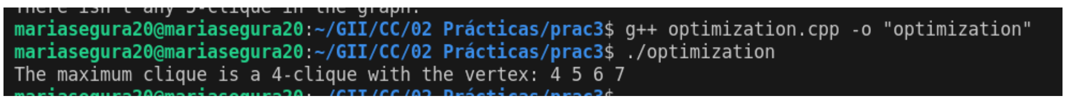
```
There isn't any 5-clique in the graph.
mariasegura20@mariasegura20:~/GII/CC/02 Prácticas/prac3$ g++ search.cpp -o "search"
mariasegura20@mariasegura20:~/GII/CC/02 Prácticas/prac3$ ./search
Size of the clique: 4
There is a 4-clique in the graph with the vertex: 4 5 6 7
mariasegura20@mariasegura20:~/GII/CC/02 Prácticas/prac3$ ./search
Size of the clique: 5
There isn't any 5-clique in the graph.
```

Figure 4: Search problem

As we checked before with the cliquer, the function returns the set (4, 5, 6, 7) as the nodes that make up the 4-clique.

3 The clique problem on its optimization version

The optimization version of the Clique problem is that, given a graph, find a maximum-size clique. In this case, I have developed a function to solve this (in optimization.hpp) that uses the binary search to find the maximum size of the clique, if it exists, and once I get the value of k, I call to the search function to get the nodes that make up that clique.



```
There isn't any 5-clique in the graph.
mariasegura20@mariasegura20:~/GII/CC/02 Prácticas/prac3$ g++ optimization.cpp -o "optimization"
mariasegura20@mariasegura20:~/GII/CC/02 Prácticas/prac3$ ./optimization
The maximum clique is a 4-clique with the vertex: 4 5 6 7
mariasegura20@mariasegura20:~/GII/CC/02 Prácticas/prac3$
```

Figure 5: Optimization problem

Bibliography

[1] GitHub Pages. *DIMACS CNF - Varisat Manual*.
URL: https://jix.github.io/varisat/manual/0.2.0/formats/dimacs.html.