# Maximum Independent Set Problem

Rafael Gomes de Sá, 104552, rafael.sa@ua.pt, MEI

***Resumo* – O presente relatório apresenta o algoritmo desenvolvido, utilizando uma abordagem probabilística, para resolver o problema do conjunto independente de vértices de cardinalidade máxima de um grafo não-orientado, assim como a análise ao esforço computacional realizado pelo algoritmo.**

**Para efetuar a análise ao esforço computacional realizado pelo algoritmo foi analisada a sua complexidade, e foi realizada uma sequência de testes com instâncias sucessivamente maiores do problema, de forma a registar e analisar um conjunto de métricas.**

**Por último, foi comparado o desempenho da estratégia probabilística com a estratégia exaustiva desenvolvida no primeiro projeto.**

***Abstract* - This report presents the algorithm developed, using a probabilistic approach, to solve the maximum independent set problem of an undirected graph, as well as the analysis of the computational effort performed by the algorithm.**

**To perform the analysis of the computational effort performed by the algorithm, its complexity was analyzed, and a sequence of tests was performed with successively larger instances of the problem, to record and analyze a set of metrics.**

**Finally, the performance of the probabilistic strategy was compared with the exhaustive strategy developed in the first project.**

## I. INTRODUCTION

A graph is a pair $G = (V, E)$, where the elements of $V$ are the vertices of the graph $G$, and the elements of $E$ are its edges. The usual way to picture a graph is by drawing a dot for each vertex and joining two of these dots by a line if the corresponding two vertices form an edge [1]. Graphs can be classified based on whether they are undirected or directed. An undirected graph, which is what will be considered in this project, simply represents edges as lines between the nodes, with no additional information about the relationship between the nodes [2]. Figure 1 illustrates an example of an undirected graph.

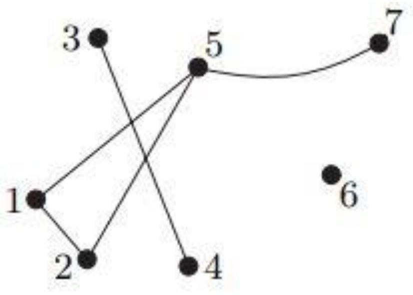

Fig. 1 - Example of an undirected graph with 7 vertices and 5 edges [1]

One of the problems that can be paraphrased to a graph problem is the maximum independent set problem. An independent set of an undirected graph is a subset of vertices in which no two vertices are adjacent. Given a set of vertices, the maximum independent set problem calls for finding the independent set of maximum cardinality [3].

A more detailed introduction on the graph theory and the problem under analysis can be found in the introduction of the report of the first project.

Following the first project, where an exhaustive search algorithm for the maximum independent set problem was developed, a randomized algorithm will be developed to solve the same problem.

A randomized algorithm is an algorithm that employs a degree of randomness as part of its logic, that is, makes random choices during its execution. Two of the benefits of this type of algorithm is simplicity and efficiency. For many applications, a randomized algorithm can be the simplest and/or the fastest algorithm [4]. As a result of this randomness, the execution time and/or the output of the algorithm may vary.

## II. ALGORITHM ANALYSIS

To analyze the computational effort performed by the algorithm, an analysis of the complexity of the algorithm must be made. To provide a better understanding, an explanation of the developed algorithm will be made first.

### *A. Graph Generation and Representation*

The generation and representation of graphs is based on the format used in the first project, with some slight changes. Only the changes will be presented in this report, since the remaining information is detailed in the report of the first project.

One of the problems detected in the version of the first project is that, in the case of graphs with a reduced percentage of edges, the distribution of the edges is not uniform along the vertices of the graph. Therefore, instead of going through the vertices orderly during the creation of the edges, the vertices are randomly selected. This results in the graph having the edges evenly distributed, even when the percentage of edges is reduced.

In addition, the generated graphs are saved in a file so they can be reused during the testing phase. In the generated file, the graphs have as header the respective number of vertices and percentage of edges.

## *B. Determination of the Maximum Independent Set*

The determination of the maximum independent set is based on the use of the sample method of the Python random module. The sample method returns a random list of k-size elements, in this case vertices, chosen from the set of vertices of the graph.

Since this is a maximization problem, it begins by generating smaller sets, starting with k = 1, and whenever one finds a k-size set that is valid, then it begins to generate random sets of size k + 1. The verification of the independence of the set is done exactly as in the first project, being that the detailed explanation is available in the corresponding report.

The algorithm continues to generate random sets until it finds a valid maximum cardinality set, or until the maximum number of attempts is reached. Initially it was intended to relate the maximum number of attempts to the total number of possible sets of the graph. However, this solution was only possible up to a certain size of the graph, since the number of possible sets grows exponentially, $2^n - 1$, and therefore the number of attempts would grow equally. For this reason, a combined approach was chosen, whereby for small-sized graphs a percentage of the total number of sets is calculated, and above a threshold the maximum number of attempts remains constant.

To avoid that the same set is tested more than once, the sets that already have been tested are stored in a list. Although repeated sets are not tested, they will count as an attempt. Initially, the goal would be to count as an attempt only those sets generated that were different from the previous ones. However, this version proved impractical, since in some cases the algorithm was not able to finish because the sample method was constantly generating repeated sets. Due to this behavior, it was decided to count all generated sets as attempts and perform a separate count for the sets that are tested.

## *C. Algorithm Complexity Analysis*

The algorithm is limited by a maximum number of attempts, $c$, and in each attempt a random set is generated which will be tested if it is not repeated. To verify the independence condition of a size $r$ set, all possible combinations of size 2 are generated, $\binom{r}{2}$.

Thus, the number of iterations that the algorithm has to perform is:

$$\leq \sum_{i=1}^{c} \frac{n(n-1)}{2}$$

where n represents the number of vertices of the graph. In this analysis it was considered that each attempt is made with the largest possible set. The verification of the repetition of the set was not considered in this analysis.

Based on this analysis, the algorithm complexity is $O(c * n^2)$. Even if the number of tested sets is not equal to the number of attempts, the complexity of the algorithm remains the same.

## III. TEST SEQUENCE ANALYSIS

To test the algorithm, a sequence of tests was performed with successively larger instances of the problem. During the tests, the execution time and the number of solutions tested were registered. In addition, the results of this algorithm were compared with the results of the exhaustive search algorithm.

To perform this analysis, tests were performed with graphs between the 2 and 25 vertices and, for each size, with 25%, 50% and 75% of edges. In these tests, the number of attempts defined was 30% of the total possible sets, with a maximum limit of 5000 attempts. In addition to these tests, tests with much larger instances of the problem will also be made, and the impact of the number of attempts will be analysed.

## *A. Number of Tested Solutions Analysis*

The results regarding the number of tested sets are shown in figure 2.

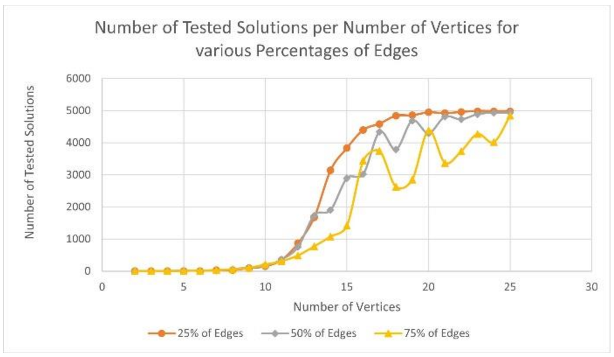

Fig. 2 - Test results regarding the number of tested solutions

Through the graphical analysis, it is possible to conclude that the number of sets that are tested grows with the size of the graph and, consequently, with the number of attempts that is assigned.

It can also be concluded that, in general, the smaller the percentage of edges of the graph, the greater the number of solutions tested. This behavior occurs, because the probability of generating a repeated set is higher for the most completed graphs. This increase in probability is because it is harder to find a valid set and, therefore, more repeated sets are generated.

## *B. Execution Time Analysis*

The results regarding the execution time of the tests performed are shown in figure 3. The execution time is given in seconds.

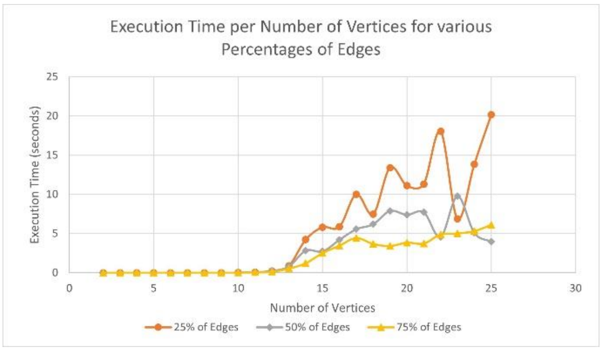

Fig. 3 - Test results regarding the execution time

By analyzing the graph, it is possible to verify that the execution time grows with the size of the graph and, consequently, with the number of sets that need to be tested. Since this value is limited, the execution time ends up stabilizing, not growing exponentially as in the case of the exhaustive search algorithm.

Furthermore, it is possible to conclude that, in general, the lower the percentage of edges, the longer the execution time. This behavior is the inverse of the one registered in the first project, because, as mentioned in subsection A, more sets are verified in the graphs with a lower percentage of edges.

## *C. Comparison of Results Between Algorithms*

To compare the results of the randomized algorithm and the exhaustive search algorithm, the cardinality of the obtained sets was analysed, and the percentage of optimal results for the randomized algorithm was calculated. The percentage was calculated by dividing the number of optimal results by the number of obtained results.

The percentages of optimal results obtained in the tests are shown in figure 4.

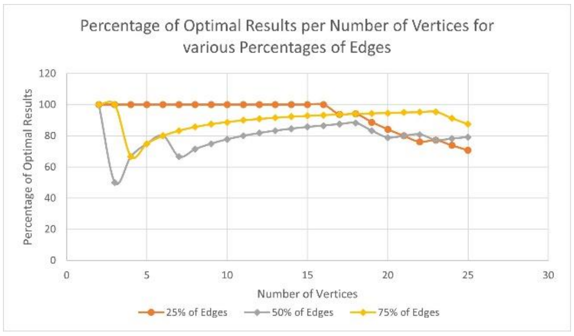

Fig. 4 - Test results regarding the percentage of optimal results

The obtained results vary widely depending on the execution, but the tendency remains the same.

Analysing the obtained results, it is possible to conclude that the greater the size of the graph, the more attempts would be necessary for the randomized algorithm to obtain the optimal solutions, since the percentage tends to decrease for graphs with more vertices.

Furthermore, the graphs with more edges tend to get the best results when analysing the graphs with more vertices. Although the graphs with more edges tend to test fewer sets, as analysed in subsection A, the cardinality of the sets also tends to be smaller and, therefore, when the attempts are over, the solution obtained is more likely to be the optimal solution.

## *D. Larger Graphs Analysis*

To analyse the results obtained with the randomized algorithm for larger graphs, tests were performed for graphs with sizes between 75 and 100 vertices, and the cardinality of the set obtained by the algorithm was recorded. The results obtained are shown in figure 5.

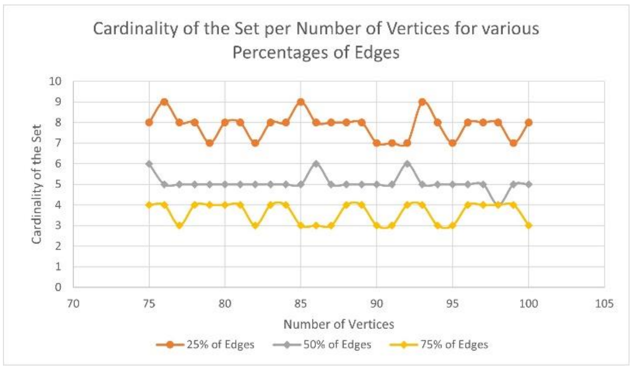

Fig. 5 - Test results with larger graphs

Analysing the obtained results, it is possible to prove an already expected conclusion, namely that the smaller the number of edges, the greater the cardinality of the obtained set.

Regarding the precision of the results, although it is not possible to verify if the obtained set is the optimal solution, the cardinality of the sets shows a small value in relation to the size of the graph. This is because the number of attempts is limited, which also limits the cardinality of the obtained sets.

## *E. Number of Attempts Analysis*

To analyse the impact of the number of attempts on the percentage of optimal results, tests were conducted with fewer attempts, since testing with more than 5000 attempts was becoming unfeasible.

Therefore, in addition to the tests with 30% of the possible sets with a limit of 5000 attempts, tests were performed with 20% of the possible sets with a limit of 1000 attempts and with 10% of the possible sets with a limit of 500 attempts. The tests were performed for graphs with 50% of edges.

The percentages of optimal results obtained in the tests are shown in figure 6.

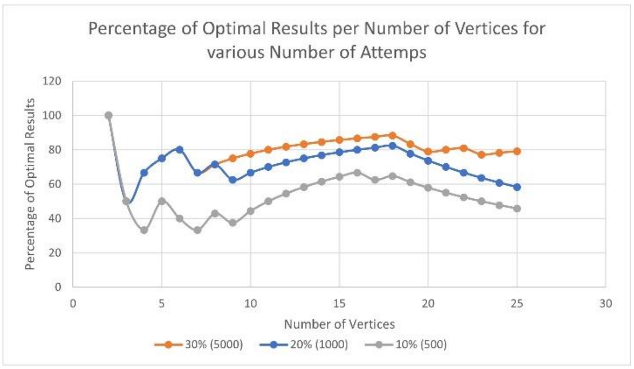

Fig. 6 - Test results regarding the number of attempts

By analysing the graphic, it is possible to conclude that the larger the number of attempts, the greater the percentage of optimal solutions. If it were possible to obtain solutions in a timely manner with a less restricted limit of attempts, the percentage of optimal solutions would get increasingly closer to 100%.

## IV. CONCLUSIONS

In short, it can be concluded that, through the randomized algorithm, it is possible to obtain results for larger graphs in a timely manner, although these are estimates and not always are optimal solutions.

Naturally, the precision of these solutions is limited by the number of attempts and sets that the algorithm tests. However, the number of attempts cannot be too large, otherwise the execution time starts to grow exponentially as in the exhaustive search algorithm.

That said, there must be a balance between the number of attempts and the precision of the results, considering the time available to execute the algorithm.

## REFERENCES

[1] Diestel, R. (2005). Graph Theory. (3rd ed.). Springer. https://doi.org/10.1007/978-3-662-53622-3

[2] Baka, B. (2017). Graphs and Other Algorithms. In B. Baka, Python Data Structures and Algorithms (pp. 168 - 190). Packt. ISBN 9781786467355

[3] Balaji, S. & Venkatasubramanian, Swaminathan & Kannan, K. (2010). A Simple Algorithm to Optimize Maximum Independent Set. 12.

[4] Motwani R., Raghavan P. (1998). An Overview of Randomized Algorithms. In: Habib M., McDiarmid C., Ramirez-Alfonsin J., Reed B. (eds) Probabilistic Methods for Algorithmic Discrete Mathematics. Algorithms and Combinatorics, vol 16. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-12788-9_3