Comparison of Time Necessary to Find The Eulerian Cycle and Hamiltonian Cycle for Graphs Depending on Number of Nodes and Saturation.

Mateusz Tabaszewski

Poznań, May 2022

Introduction and Methods of Research

This paper is meant to show the conclusions and observations regarding the time necessary to find the Eulerian Cycle and Hamiltonian Cycle depending on the number of nodes and the graph’s saturation. For this experiment graphs of different n(number of nodes) and saturations were created, with saturations equal to {0.2, 0.3, 0.4, 0.6, 0.8, 0.95}.The graphs were created by adding n edges and then filling the graph with edges until specified saturation was reached. Later an algorithm was run, which made sure that the graph was connected, by checking if every node can be reached from any other. After creation of the graphs, time necessary for finding the Eulerian Cycle and the Hamiltonian Cycle was measured using using GetProcessTimes(), GetCurrentProcess() functions included via <processthreadapi.h> header. The time it took to find either cycle was measured by taking an average of multiple measurements for multiple graphs of the same saturation and number of nodes. For the experiment involving the search for Hamiltonian Cycle, a timeout mechanism was implemented, to make sure the algorithm doesn’t get “stuck”. The timeout mechanism would trigger after 300 seconds of work on a single graph, and in response, it would stop further attempts of finding a cycle for this graph. The problem of finding the Eulerian Cycle was solved using Hierholzer’s algorithm and the problem of finding the Hamiltonian Cycle was solved using a Backtracking algorithm.

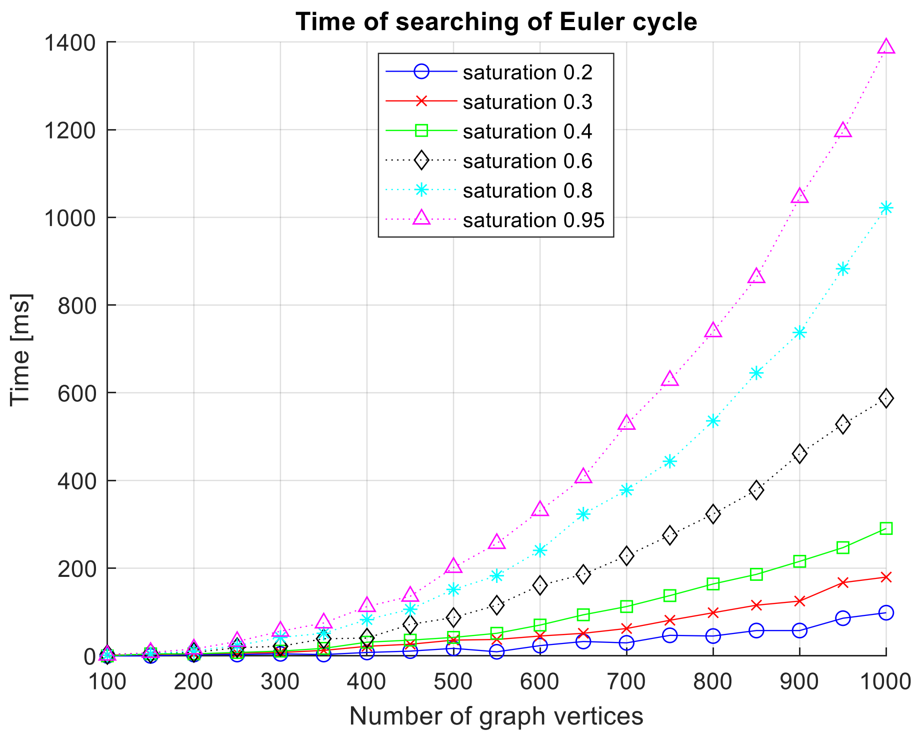

Fig.1. Time necessary to find Euler Cycle depending on saturation and number of nodes.

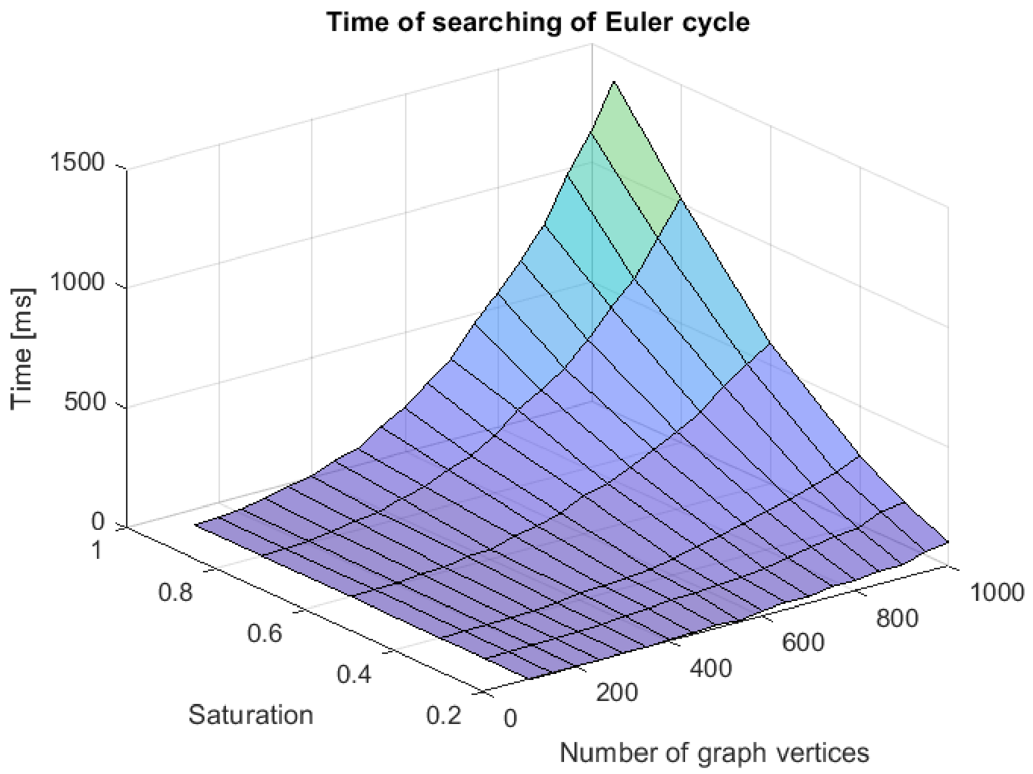

Fig.2. 3D plot, showing the time necessary to find an Euler Cycle based on number of nodes in a graph and saturation of the graph.

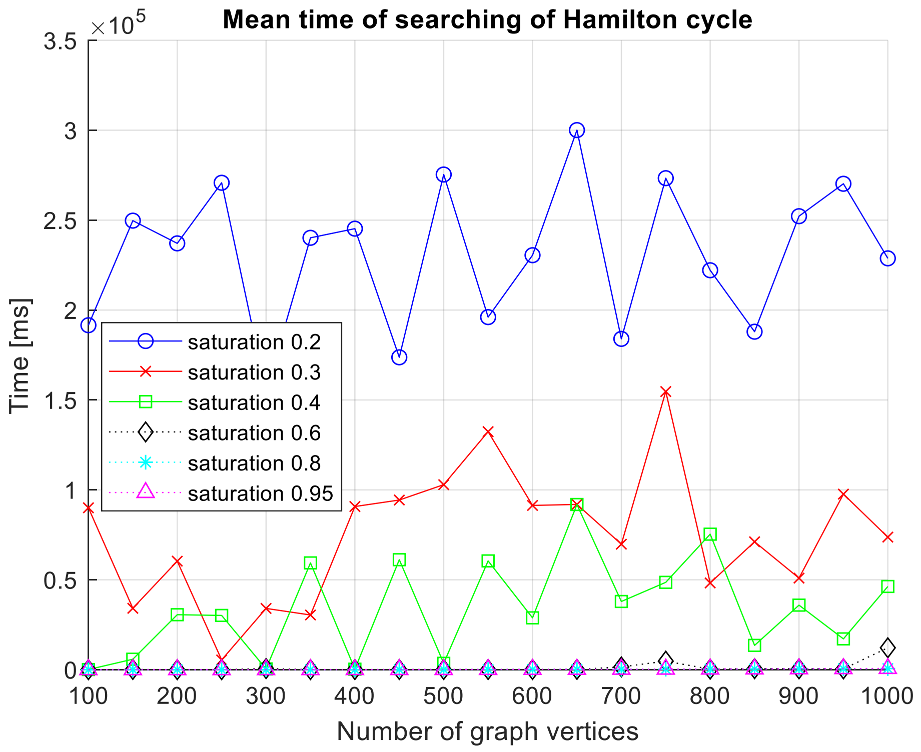

Fig. 3. Time necessary to find Hamilton Cycle depending on number of vertices in a graph and saturation of the graph (please, note that time here is measured in ms*10^5 to stay consistent with previous graphs).

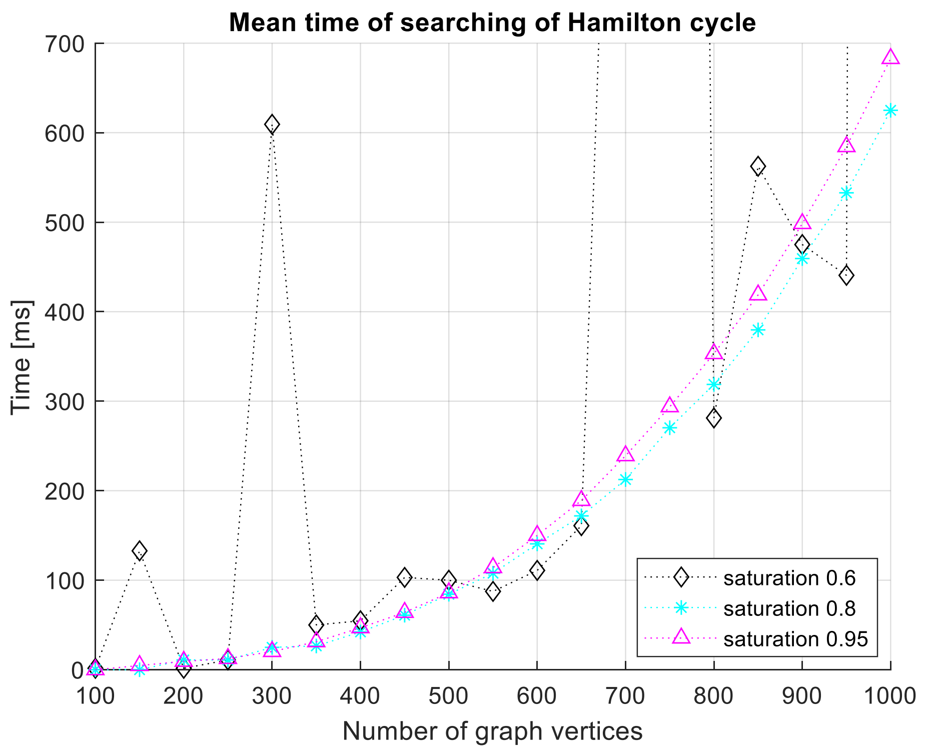

Fig.4. Zoomed in data from Figure 3, showing time for graphs of saturations={0.6,0.8,0.95}.

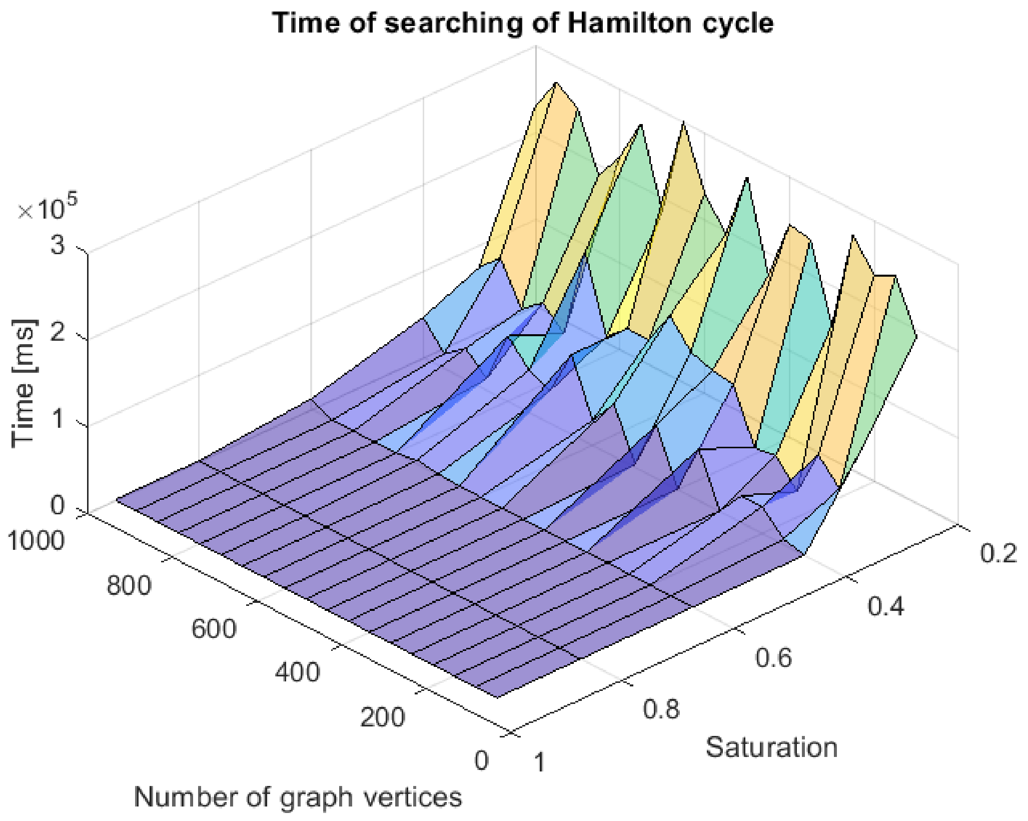

Fig.5. 3D plot showing time necessary to find the Hamilton Cycle depending on number of vertices and saturation (please, note the decreasing saturation axis).

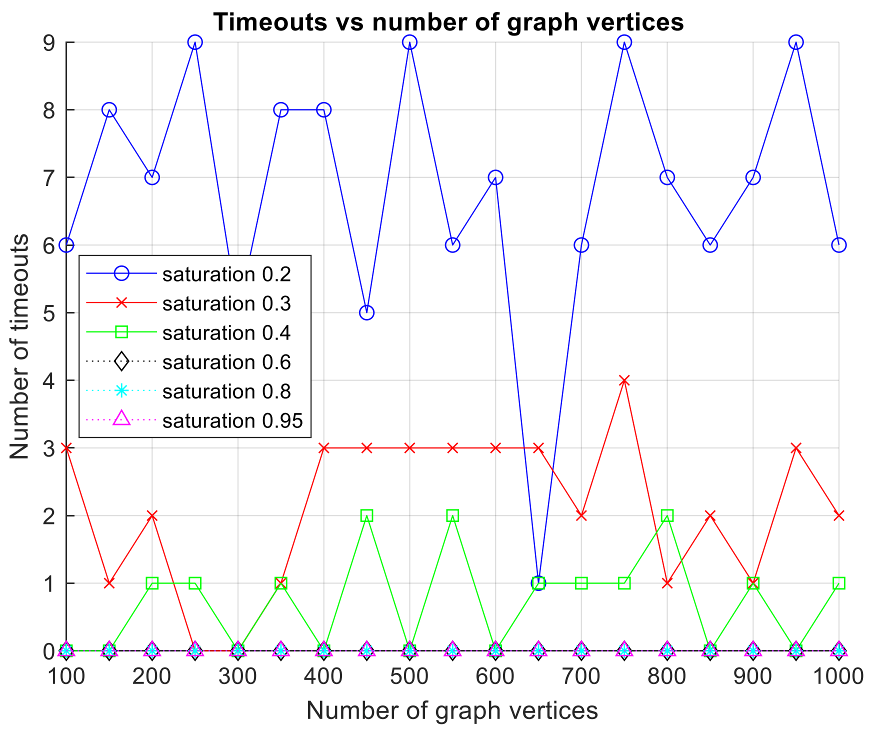

Fig.6. Number of times a “timeout” occurred, depending on saturation and number of vertices.

Observations and Conclusions

For both experiments, the method of graph representation chosen was **Adjacency List**. This choice was based on the previous project in which 4 methods of representation were chosen. Out of the tested methods, this one had the second-fastest time when it came to checking a connection between two nodes. This method is also quite memory-efficient and is also characterized by the relative ease of implementation.

As can be seen in Figures 1 and 2, the time necessary to find the Eulerian Cycle is dependent on both the number of vertices as well as saturation. As is shown on the graphs, the greater the saturation and the bigger the number of vertices, the more time it will take for the algorithm to find an Euler Cycle. As such it can be inferred that in this method of finding an Eulerian Cycle, the **time complexity is dependent on the number of edges of the graph**. This would align with graphs plotted in Figure 1, as they seem to align with a quadratic function. The number of edges in a graph is equal to $E=k*n(n-1)/2$. As such it makes sense that if the method is dependent on the number of Edges, the graph would have a shape similar to that of a quadratic function. As such it can be concluded that the time complexity of this method can be written as $O(|V+E|)$. And As such, it can be concluded that the problem of finding The Eulerian Cycle is an NP-complete problem.

As can be seen in Figures 3,4,5 generally, the time necessary to find the Hamiltonian cycle seems to be longer for smaller saturations. As per Figure 6, the number of timeouts is also bigger for smaller saturations. However, there is also a lot of variation in the time it takes to find the cycle, even for the same saturations. As such, it seems that the time is highly dependent on the particular graph, and it can vary highly even for different graph with the same number of vertices and saturations. However, it seems that for big saturations=0.95 and 0.8 the variations are less noticeable and the time just rises with number of vertices in a more "predictable" manner. There is also a relatively small difference between times for k=0.8 and k=0.95 as opposed to the differences between the remaining saturations. The time complexity of this algorithm is O(n!), which explains, to a certain extent, the variations between seemingly similar graphs.