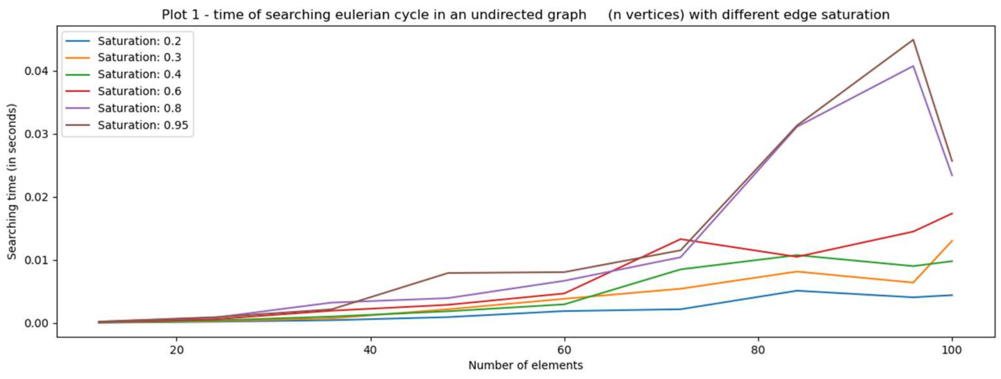
Rafał Stachowiak conclusions

Eulerian cycle:

I have chosen adjacency list as a graph representation because it has low complexity and it was easy to implement it for me. Other representations would be good also (apart from adjacency matrix which is bad graph representation of high complexity)

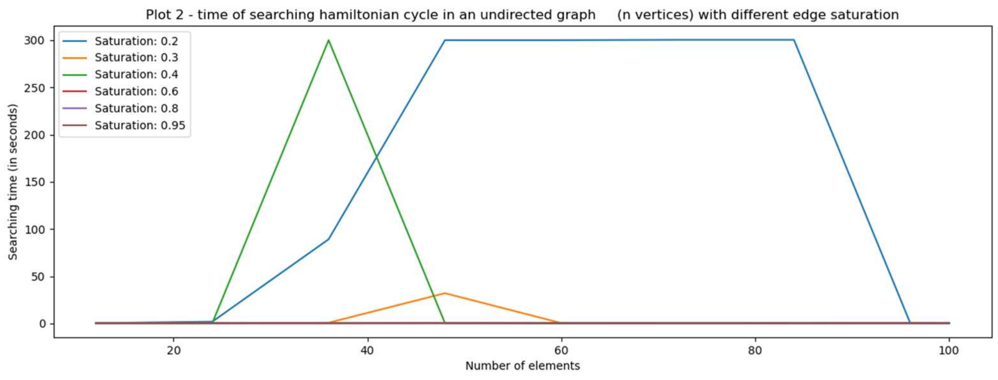
The complexity of searching eulerian cycle in the graph is O(E) (Hielholzer algorithm) so it is effective, and fast. Of course the higher the saturation of edges, the more time is needed to perform computations but it is still faster than Hamiltonian Cycle algorithm.



Hamiltonian cycle:

I have used backtracking algorithm to implement searching Hamiltonian cycle in the graph. It is fast when the saturation of edges is high, but it takes much time to look for the cycle when the saturation is low (because nearly all paths must be checked and they have little amount of branches)

When it comes to complexity, in a recursive call of function looking at the worst case scenario, every function call decreases the branch number by 1 so we have: n, then (n-1) (n-2) which leads to complexity O(n!) which is really high and that’s why in my program there is a time boundary of 300 sec.


Plot 2 - time of searching hamiltonian cycle in an undirected graph (n vertices) with different edge saturation
Saturation: 0.2
Saturation: 0.3
Saturation: 0.4
Saturation: 0.6
Saturation: 0.8
Saturation: 0.95
Searching time (in seconds)
Number of elements